



**Abertay
University**

Report of Somestore.com/Rickstore Penetration Test

Christian Hegarty

CMP509: Ethical Hacking

2021/22

Abstract

This paper aims to outline the methodology followed and results discovered during a penetration test of the website “Rickstore”, and its admin pages named “Somestore”. The penetration test utilized the OWASP v4 framework to test, analyse, and critically evaluate the security of this website. This was carried out as a black box test with no access to the virtual machine the server is hosted on.

This utilized all 11 classes of test defined by OWASP, ranging from general information gathering to authentication testing and client-side testing

The website was found to have severe security issues with several complete exploits that would allow attackers to gain access to admin control of the website without even having a customer account. These exploits include SQL injection, cross site scripting and authentication bypassing.

Outlined in this paper are several solutions to some of the most critical security flaws found during testing.

Contents

1	Introduction	1
1.1	Background	1
1.2	Aim	2
2	Procedure.....	3
2.1	Overview of Procedure	3
2.2	Information Gathering	4
2.2.1	Fingerprint Web Server.....	4
2.2.2	Review Web Server Metafiles for Information Leakage	4
2.2.3	Enumerate Applicatitons on Webserver.....	4
2.2.4	Review Webpage Comments and Metadata for Information Leakage	5
2.2.5	Identify Application Entry Points	5
2.2.6	Fingerprint Web Application Framework	5
2.2.7	Map Application Architecture.....	5
2.3	Configuration and Deploy Management Testing.....	6
2.3.1	Test Application Platform Configuration	6
2.3.2	Test File Extensions Handling for Sensitive Information.....	6
2.3.3	Backup and Unreferenced Files for Sensitive Information	6
2.3.4	Enumerate Infrastructure and Application Admin Interfaces	7
2.3.5	Test HTTP methods	7
2.3.6	Test HTTP Strict Transport Security	8
2.3.7	Test RIA Cross Domain Policy.....	8
2.4	Identity management testing	8
2.4.1	Test Role Definitions	8
2.4.2	Test Account Provisioning Process.....	9
2.4.3	Testing for Account Enumeration and Guessable User Account.....	9
2.4.4	Testing for Weak or Unenforced Username Policy.....	9
2.5	Authentication testing	10
2.5.1	Testing for default credentials	10
2.5.2	Testing for Weak lock out mechanism.....	10
2.5.3	Testing for bypassing authentication schema	10
2.5.4	Test remember password functionality.....	11

2.5.5	Testing for Browser cache weakness	11
2.5.6	Testing for Weak password policy	12
2.5.7	Testing for weak password change or reset functionalities	12
2.6	Authorisation testing	12
2.6.1	Testing Directory traversal/file include.....	12
2.6.2	Testing for bypassing authorization schema	13
2.6.3	Testing for Privilege Escalation	13
2.7	Session management testing	14
2.7.1	Testing for Bypassing Session Management Schema	14
2.7.2	Testing for Cookies attributes.....	14
2.7.3	Testing for Cross Site Request Forgery	14
2.7.4	Testing for logout functionality.....	15
2.7.5	Test Session Timeout	15
2.8	Data validation testing	15
2.8.1	Testing for Reflected Cross Site Scripting	15
2.8.2	Testing for Stored Cross Site Scripting	16
2.8.3	Testing for SQL Injection	16
2.8.4	MySQL Testing.....	17
2.9	Error Handling	17
2.9.1	Analysis of Error Codes	17
2.10	Cryptography	18
2.10.1	Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection	18
2.11	Business logic testing	18
2.11.1	Test Upload of Unexpected File Types.....	18
2.12	Client side testing.....	19
2.12.1	Testing for Clickjacking.....	19
2.12.2	Testing WebSockets	19
3	Discussion.....	20
3.1	General Discussion.....	20
3.2	Countermeasures.....	20
3.3	Future Work	20
References	22
Appendices.....		23

Appendix A 23

Appendix B **Error! Bookmark not defined.**

Appendix C - Suggestions for formatting figures/tables/screenshots in the body of the text..... **Error!**
Bookmark not defined.

1 INTRODUCTION

1.1 BACKGROUND

Penetration testing is the process of analysing the security of IT systems by attempting to breach some or all of the systems security by using the same tools an attacker would have access to. It can be thought of similar to an audit, where an external group ensures that the security process being used for a system are sufficient.

Penetration testing can be carried out with different degrees of already known information about the system. Depending on the amount of information given this can be described as white-box, grey-box, or black-box testing. White-box testing describes the type of test where all information about the system being tested is shared with the tester. This type of test is mostly used if there is already an internal vulnerability assessment or management controls are already in place and would be used to verify how well these systems are working by identifying any known vulnerabilities or misconfigurations that have not been flagged by the systems already set up. Given that Somestore does not have any of these systems in place, this type of test would not be ideal.

Black-box testing is a type of penetration test carried out in which the internal structure of the system is not given to the tester. This practice allows the tester to better simulate an attack from an external party, who wouldn't know anything about the internals of the system without probing it. While this means that it more accurately models the risk a system might face in practice, it also carries the downside of the tester potentially missing vulnerabilities due to lack of information. This will be the type of test carried out for Somestore as it will provide the owner with a more accurate list of real-world exploits that could be carried out by external parties.

Grey-box testing is a type of testing in which the internal structure of the system is only partially known. This means that while the source code is not known, knowledge of the of any high-level program behaviour such as its architecture and internal states are shared. Tests are then designed based on this knowledge.

Somestore is an online shop currently used to sell electronics. It was developed by an external web development company and has now been bought by a new owner. The owner has noticed that the website contains several bugs and is concerned about the possibility of security flaws caused by these bugs which may be used to hack the application. The owner has contracted a penetration tester to carry out a test of the web application to find potential security flaws and exploits. The owner expects the report to contain the findings of the test and recommendations of what to do in order to make the website more secure.

This report will include information on the methodology selected and why it is selected. Detailed information of the procedure will be described in order to allow external testers to replicate and corroborate the results of the test. Detailed and readable results which are understandable to other testers should also be included as well as information and advice on how to use these results to create a more secure version of the website being tested.

This test and report are important parts of maintaining security. Data from “Accenture” shows that the average cost of cyberattacks have been increasing between 2017 and 2018 the average cost of web-based attacks has risen 13% to an average of \$2.28 million (Accenture, 2019). This can be seen in figure 1.

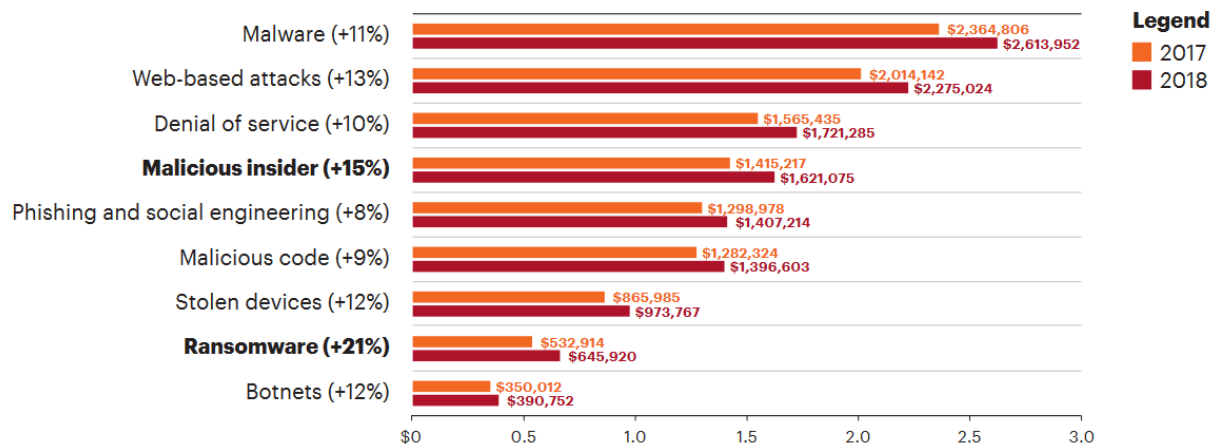


Figure 1: Accenture data of cost of attacks between 2017-2018. Courtesy of: <https://www.accenture.com/acnmedia/PDF-99/Accenture-Cost-Cyber-Crime-Infographic.pdf#zoom=50>

This means it is very important web-based businesses do something to become more secure. A Penetration test is a very good way of achieving this.

1.2 Aim

The aim of this paper is to describe the process utilized for the penetration test of Somestore.com in such a way that it is replicable by another penetration tester. It also attempts to display the results in an easily understandable way so that the owners of Somestore have some understanding of any security risks found. There should also be some description of how to fix or mitigate some of the security risks found during the test.

Essentially, the paper should outline the following topics in the following order:

- Description of methodology
- Detailed description of how tests were carried out and the results of each test
- Discussion of discovered security flaws
- Potential fixes

2 PROCEDURE

2.1 OVERVIEW OF PROCEDURE

It was decided that the OWASP v4 methodology should be used to carry out the pen-test. This methodology was chosen for it.

Using the OWASP methodology helps identify key vulnerabilities listed in the OWASP top ten vulnerabilities. These are:

- Broken access control
- Cryptographic failures
- Injection
- Insecure design
- Security misconfiguration
- Vulnerable and outdated components
- Identification and authentication failures
- Software and data integrity failures
- Security logging and monitoring failures
- Server-side request forgery

In order to test and discover the vulnerabilities in this list. All categories of test featured in the methodology will be tested. These categories are:

- Information Gathering
- Configuration and Deploy Management Testing
- Identity Management Testing
- Authentication Testing
- Authorization Testing
- Session Management Testing
- Data Validation Testing
- Error Handling
- Cryptography
- Business logic Testing
- Client-Side Testing

The tools used throughout the testing process are easily accessible in order to make the test as replicable as possible. The bulk of testing was carried out using OWASP ZAP and OWASP Mantra. Several smaller tools were used in various tests. Each test will provide a description of the procedure, including what tool was used and how it has been used to allow others to verify the results. These tools are all accessible online or via the Kali Linux suite.

2.2 INFORMATION GATHERING

2.2.1 Fingerprint Web Server

Fingerprinting the web server is very important to the overall penetration test as knowing the type of web server and its version will allow the tester to determine any known vulnerabilities associated with the specific web server version. It also allows the user to choose more appropriate exploits to try during later testing.

This test was carried out using HTTPPrint. Which reported the server banner found in the HTTP response header.

Result:

The server banner indicated that the web server used Apache 2.4.29. this is out of date and has several vulnerabilities associated with it. Other technologies used in the website are OpenSSL version 1.0.2n, PHP version 5.6.34, mod_perl version 2.0.8-dev and Perl version 5.16.3. several of these technologies are also out of date and should be updated to a newer version immediately due to the risk of security vulnerabilities.

2.2.2 Review Web Server Metafiles for Information Leakage

This test reviews the “robots.txt” file within the web server for any information leakage. This is carried out by manually looking at the file to find any information on the server’s directories or to find any disallowed files which may be useful for fingerprinting the web application

Result:

Analysis of “robots.txt” shows little information. The only thing disallowed is the file “schema.sql”, indicating an SQL server present in the web server. This result can be seen in appendix 1.

2.2.3 Enumerate Applications on Webserver

It is incredibly important to discover which applications are being hosted on the server being tested. Applications can be very likely to have known vulnerabilities which can be used to exploit the whole server. This test was carried out using Nmap to search for non-standard ports which are open. This was done using the command “nmap -p 1-65535 -sT 192.168.1.20”

Result:

4 ports were found to be open. Port 21 for ftp, 80 for http, 443 for https, and 3306 for MySQL. The first 3 ports are relatively standard and only really indicates the use of the HTTPS protocol which should make the website more secure. 3306 is a non-standard port. This is used for MySQL to communicate. This indicates that the website uses a SQL database in its backend and uses MySQL to interact with the database. This can be seen in appendix 2.

2.2.4 Review Webpage Comments and Metadata for Information Leakage

Detailed comments are sometimes left in HTML code. This may reveal internal information about the web server such as admin details, IP addresses or SQL code. This is carried out by manually looking at HTML source code in each web page for any comments which might contain this sensitive information.

Result:

Most web pages had few detailed comments. The only large comment found was in the index page. It described a code snippet to fix a viewport bug. Overall, there does not appear to be any information leakage within webpage comments.

2.2.5 Identify Application Entry Points

This test aims to help identify and map out the website in order to find areas which should be investigated after information gathering has been completed. These areas can be parts of the website where GET and POST requests are used along with fields in the webpage which users can interact with which are used in the POST requests such as text boxes. This was carried out by spidering the web server using OWASP ZAP. This was also used to map the execution paths through the application.

Result:

Spidering the web server provided a collection of all GET and POST requests in the website along with all fields used in these requests. This is valuable information which can later be used for various other tests such as Data Validation testing. Principal workflows were also discovered. This can be seen in appendix 3.

2.2.6 Fingerprint Web Application Framework

This procedure will provide information on the vendor and version of the web framework being used by the web server. This could potentially show vulnerabilities or misconfigurations which may help in the process of carrying out exploits. Whatweb was used to search for this information.

Result:

The results can be seen in figure 2. It can be seen that the web framework that the website uses is PHP. This particular framework can be insecure if misconfigured. The version of PHP being run is 5.6.34. several vulnerabilities have been found in this version, these can be found in the CVE database (PHP : List of security vulnerabilities, n.d.), however, this version has reached end of life so there are no more security updates so any future vulnerabilities may not be fixed. The website also uses the client-side framework, jQuery version 1.6.2. This is also an out-of-date framework. Several vulnerabilities cross site scripting can be found in the CVE database (jQuery : List of security vulnerabilities, 2022).

```
root@kali:~/Desktop# whatweb -a 3 192.168.1.20
http://192.168.1.20 [200 OK] Apache[2.4.29][mod_perl/2.0.8-dev], Cookies[PHPSESSID], Country[RESERVED][22], HTTPServer[Unix][Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3], IP[192.168.1.20], JQuery[1.6.2], OpenSSL[1.0.2n], PHP[5.6.34], Perl[5.16.3], Script[text/javascript], Title[RickStore Groups], X-Powered-By[PHP/5.6.34]
```

figure 2: Whatweb analysis of Somestore

2.2.7 Map Application Architecture

Knowing what type and version of web server is being used allows testers to find previously known vulnerabilities and appropriate exploits that could be carried out during testing. This information is generally found by sending the web server commands and analysing output. Different types of web

servers respond to these commands in different ways, making it possible to discern the type and version of web server being used by the website. This was tested along with fingerprinting the web application framework by using Whatweb.

Result:

The results of this procedure can also be seen in figure 2. It can be seen that the website uses Apache. The version being used (2.4.29), is out of date. Once again, there are several vulnerabilities associated with this version which can be found in the CVE database [xxx].

2.3 CONFIGURATION AND DEPLOY MANAGEMENT TESTING

2.3.1 Test Application Platform Configuration

Misconfigured elements of the web server could compromise the security of the entire application. This is why it is important to test if there are any leftover default files such as documentation or test pages which should be removed to avoid exploitation. To find any default files for this website, Nikto was used using the command “nikto -h 192.168.1.20” the results of this tool can be seen in figure 2.

Result:

4 default files were found. Phpinfo.php, config.php, info.php, and preview.php. config.php is completely empty and can be ignored. “preview.php” appears to be broken but Nikto says it may include a remote file inclusion vulnerability. “phpinfo.php” and “info.php” contain information about the installation of the web server. Nikto output can be seen in appendix 4.

2.3.2 Test File Extensions Handling for Sensitive Information

This test was carried out using Nikto. This test was carried out simultaneously with the same command as the previous test.

Result:

Dirbuster found several external files. Most were not useful as they were just to do with external fonts used for the website, however schema.sql was found which could prove useful for further exploitation. Nikto output can be seen in appendix 4.

2.3.3 Backup and Unreferenced Files for Sensitive Information

This test was carried out using Dirbuster to carry out a list based brute force enumeration on the website. The target URL was set to 192.168.1.20, the dictionary file was “directory-list-2.3-small.txt”, starting options were all default and file extensions being searched for were “.old”, “.bak”, “.inc”, and “.src”.

Result:

“sqlcm.bak” was found by Dirbuster and it was manually analysed. It was found to contain a code

snippet of a script alerting the user of the website filtering input due to a hacking attempt. This suggested there is some amount of defence against some form of data input attack such as an SQL injection attack or cross site scripting attack. The contents of the backup file can be seen in figure 3.

figure 3: Contents of sqlcm.bak

2.3.4 Enumerate Infrastructure and Application Admin Interfaces

This test also used Dirbuster. The result of the previous test was applicable to this test as well.

Result:

Dirbuster found the location of admin pages. These are located in the directory “/admin”. Dirbuster results can be seen in appendix 5.

2.3.5 Test HTTP methods

HTTP methods were tested using nmap. This was carried out using the command “nmap --script http-methods 192.168.1.20” this provides more information than necessary but displays the http and https methods used in the website.

Result:

the results of the nmap script showed that there are 4 allowed http options over http: "GET HEAD POST OPTIONS", and 5 options allowed over https "POST OPTIONS HEAD GET TRACE". TRACE is often used for debugging purposes as it echoes back a string that is sent to the server; however, it should be removed once the website has completed development as it is potentially risky. It can be used to carry out a type of attack called cross site tracing (Grossman, 2003). The result of the nmap script can be seen in figure 4.

figure 4: HTTP options from nmap script

2.3.6 Test HTTP Strict Transport Security

This test was carried out using curl. The command used to get the result was a modification of the command given in the OWASP checklist “curl -s -D- https://domain.com/ | grep Strict” displays nothing. Removing the grep portion and the silent flag “-s” shows that the command does not work because the domain is untrusted, therefore the flag –insecure must be used. Due to this the command used for this test was “curl --insecure -D- https://192.168.1.20/ | grep Strict”

Result:

No HSTS header could be found from using the curl command. Due to this it is likely that the website does not use the HTTPS protocol despite having it set up.

2.3.7 Test RIA Cross Domain Policy

This test was carried out by manually searching for the files “crossdomain.xml” and “clientaccesspolicy.xml”

Result:

No policy files could be found suggesting that there are no RIA programs being used in this website. This test is therefore not applicable.

2.4 IDENTITY MANAGEMENT TESTING

2.4.1 Test Role Definitions

This test was carried out manually, the abilities of 3 user groups were tested and entered into a permission matrix which can be found in figure 5.

	No account	Customers	Employees
buy products			
Create Customer account			
use contact page			
view products			
log in			
all other functionality in rickstore			
access to somestore admin pages			

figure 5: permission matrix of users of Somestore

Result:

All employees have access to all aspects of the admin page, customers only have access to the main website. This violates the principle of least privilege and means that if an attacker gains access to any employees account, the attacker will have access to the entire website. This should be immediately fixed, or a policy of least privilege should be implemented within the business.

Non-account holders can do much of what customer accounts can do except for aspects of stored personal details and log out functionality and have access to the login and create account page which account holders cannot do while logged in.

2.4.2 Test Account Provisioning Process

This test was done by manually creating accounts and looking for permissions required to do so

Result:

It was found that any employee account can create other employee accounts. This is similar to the previous test where a policy of least privilege will likely mitigate much of this risk. It would be advisable that only HR staff should be able to provision accounts.

2.4.3 Testing for Account Enumeration and Guessable User Account

This was tested manually by attempting to log in to both the customer login page and admin page using 3 different types of attempts. One attempt with an incorrect username and password, one with a correct username and incorrect password, and one attempt with an incorrect username and correct password, leading to a total of 6 tests.

Result:

It was discovered that the website's response to login attempts differs depending on how many fields are correct. If the username is incorrect, an alert pops up saying the username is invalid, whereas, if a real username is used with an incorrect password, it refreshes the page immediately with empty fields. Due to this and the absence of a lockout mechanism, it is possible to enumerate the username of customers by brute forcing until the alert pops up. This is not possible for admin accounts as attempting with a correct username and incorrect password as well as both incorrect leads to the same error 404 page. Tests featuring a correct password and incorrect username behaved the same as tests with both an incorrect password and incorrect username. Results of this can be found in appendix 6-11

2.4.4 Testing for Weak or Unenforced Username Policy

This test was carried out by creating customer accounts and employee accounts with weak usernames, such as single letters, email addresses and already made accounts.

Result:

Customer accounts require a username in the form of an email address. "test", "test.com", and "test@.com" all failed, suggesting the field is looking for an input in the format "[name]@[email].com" is required. Interestingly, the username "@test.com" appeared to fail when testing, however upon inspection of the customer table, a record of that account can be found. Meaning "[name]" is not needed to create a customer account.

There is no visible username policy that is for the add employee page. Usernames associated with previously made accounts can be used for newly made accounts for both customers and employees.

2.5 AUTHENTICATION TESTING

2.5.1 Testing for default credentials

This was tested manually by attempting to leave usernames or passwords blank during the creation of accounts. This was tested on both employee and customer accounts.

Result:

Default of every field for newly created employees is " " if nothing is entered in each field during registration. This can be seen in figure 6. Only the employee image requires a file to be selected. This does not happen for customer accounts as it is not possible to leave the email or password fields blank.



<input type="checkbox"/>	53	testadmin	testadmin	testadmin		 
<input type="checkbox"/>	55					 

figure 6: blank employee details compared to correctly filled in employee

2.5.2 Testing for Weak lock out mechanism

This was done by manually failing login attempts multiple times on purpose for both employees and customers.

Result:

There is no lockout mechanism on customer logins. This means that the login function is vulnerable to brute force/dictionary attacks. Testing was unable to conclude if the employee login page was similarly vulnerable as incorrect entry of accounts led to an error 404.

2.5.3 Testing for bypassing authentication schema

Several sub-tests were carried out for this test. Initially brute forcing was attempted to access the employee pages. This was carried out using the URL

["http://192.168.1.20/admin/index.php?authenticated=yes"](http://192.168.1.20/admin/index.php?authenticated=yes). Parameter modification was also attempted. This was carried out using OWASP ZAP to modify the GET header from "GET http://192.168.1.20/admin/index.php HTTP/1.1" to "GET http://192.168.1.20/admin/index.php?authenticated=yes HTTP/1.1".

SQL injection was also carried out, however this will be mentioned in the SQL injection test section

Result:

Testing via the brute forcing method and parameter modification proved inconclusive. No pages were able to be bypassed using any of these methods. Bypassing authentication was possible using SQL injection. This will be explained in detail in the SQL injection testing section.

Reattempting the brute forcing method and parameter modification method proved successful. it is believed that the first attempts failed due to a customer account already being logged in. if a customer is not logged in, it is possible to bypass the authentication and gain access to all of the employee pages

2.5.4 Test remember password functionality

Remember password functionality testing was carried out manually by checking for differences in cookies when the remember password checkbox was selected or deselected. This was done for both customer and employee accounts.

Result:

The remember password checkbox did not appear to be functional. Usernames and passwords were always remembered for customer logins and never remembered for employee logins whether the checkbox was selected or not. If the customer login is taken as the remember password functionality being used, it can be seen that a new cookie is created called “SecretCookie”. This is encoded but can easily be decoded in 2 steps. First by converting from base64 followed by a Caesar cipher where a=n, b=o, c=p, etc. the decoded cookie reads as “[email address]:[password]:[Unix timestamp]”. It is also important to note that this cookie expires at the end of the session, so it doesn’t even remember the account for longer than if it were not selected at all. An example of a decoded SecretCookie can be found in figure 7.

The screenshot displays a web application interface for decoding data. On the left, a 'Recipe' sidebar lists various tools: 'URL Decode', 'From Base64', and 'Substitute'. The 'From Base64' tool is selected, showing a dropdown for 'Alphabet' set to 'A-Za-z0-9+/' and a checked option for 'Remove non-alphabet chars'. The 'Substitute' tool shows a mapping from 'Plaintext' (abcdefghijklmnopqrstuvwxyz) to 'Ciphertext' (NOPQRSTUVWXYZABCDEFGHIJKLM). The main area is split into 'Input' and 'Output' sections. The 'Input' section shows a base64-encoded string: 'Z3JmZ0BncmZnLnBiejpncmZnOjE2NTE1MDc3OTE%3D'. The 'Output' section shows the decoded result: 'TEST@TEST.COM:TEST:1651507791'. Metadata for the input and output is also visible, such as 'length: 42' and 'lines: 1' for the input, and 'time: 0ms', 'length: 29', and 'lines: 1' for the output.

figure 7: decoding of secret cookie

2.5.5 Testing for Browser cache weakness

Browser cache weakness was tested by logging into accounts, navigating to a page, logging out of the account and trying to return to the previous page. This was carried out for both employee and customer accounts.

Result:

Employee pages are accessible after logging out by returning to the last page, but the employee account is not remembered. The website contains cache control for customer pages and no test showed any ability to return to a logged in state after going back a page.

2.5.6 Testing for Weak password policy

This was tested by creating accounts with different types of weak passwords such as single characters or changing passwords of already made accounts to test if reused passwords are allowed.

Result:

The website appears to have almost no enforced password policy. In testing it was found that a password could be as short as "t" (seen in figure 8 and resetting a password allowed the customer to continuously re-use the same password. Employee accounts can be created with blank passwords which default to "".

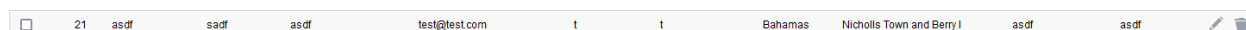


figure 8: customer with password "t"

2.5.7 Testing for weak password change or reset functionalities

This was tested by resetting passwords of accounts and manually looking at what was required. Some password reset attempts left fields blank to see if any and all fields are needed to reset the password

Result:

The password reset functionality appears to be very weak there is no external authentication such as an email with a link to reset. It can be reset immediately via the user account page. The page appears to require the old password to change to a new one, however this can be left blank or filled with anything as it the password can be reset no matter what is in the old password field. Passwords are displayed in plaintext and the page is potentially vulnerable to CSRF attacks as the page is missing anti-CSRF tokens. This cannot be confirmed as it was not possible to exploit a CSRF attack on the website. New password and re-enter password field need to match for the password to be reset.

2.6 AUTHORISATION TESTING

2.6.1 Testing Directory traversal/file include

This was carried out by running an active scan on OWASP ZAP.

Result:

OWASP ZAP displayed a high alert flag for path traversal for this website in the page attachment.php. opening the URL displays a list of directories and files on the computer hosting the website. This list can be seen in figure 9.

2.7 SESSION MANAGEMENT TESTING

2.7.1 Testing for Bypassing Session Management Schema

This test was carried out by reading the cookies in “Cookies Manager+”, copying the content and inputting the content into “CyberChef”. This was used to attempt to decode the content of the cookie. During decoding several pipelines were used, however, only the successful one was recorded.

Result:

The session ID does not appear to be easily decryptable, attempted various decoding methods such as base64 and hex code, however there is no noticeable pattern. The secret cookie on the other hand is decryptable. This is explained in the remember password test.

2.7.2 Testing for Cookies attributes

The attributes of the two different cookies were read using “Cookies Manager+”

Result:

Cookies are sent over any connection and are not HTTPOnly. All cookies expire at the end of the session, rendering the remember password checkbox non-functional.

2.7.3 Testing for Cross Site Request Forgery

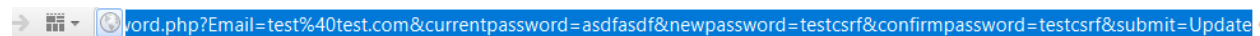
CSRF attacks were tested by looking at the pages OWASP ZAP discovered had no anti-CSRF tokens. A test was carried out to attempt to reset a user’s password using the URL

“192.168.1.20/updatepassword.php?Email=test%40test.com¤tpassword=asdfasdf&newpassword=testcsrf&confirmpassword=testcsrf&submit=Update”.

Result:

There is an absence of any anti-CSRF tokens in almost every page on the website, meaning the website is potentially vulnerable to this kind of attack. An attempt was made to reset a password using a CSRF attack using the URL

“192.168.1.20/updatepassword.php?Email=test%40test.com¤tpassword=asdfasdf&newpassword=testcsrf&confirmpassword=testcsrf&submit=Update”, however, this did not work. Based on the error it may be due to a problem with the user’s session ID. The error caused by this failed attempt can be seen in figure 10.

A screenshot of a web browser's address bar. The URL is partially visible and highlighted in blue: "ord.php?Email=test%40test.com¤tpassword=asdfasdf&newpassword=testcsrf&confirmpassword=testcsrf&submit=Update". The browser interface shows standard navigation buttons (back, forward, home, search) and a search engine icon.

Notice: Undefined index: thumbnail in /opt/lampp/htdocs/studentsite/usersession.php on line 8

figure 10: result of CSRF password reset attack

2.7.4 Testing for logout functionality

This was carried out by logging into an account, copying the session ID and Secret Cookie, logging out and attempting to get into the accounts user details by modifying the POST request to include the cookies.

Result:

Logging out of an account appears to work correctly. An attempt was made to get into a customer's account details after logging out by editing the HTTP header and changing the session ID, but it just returned to the index page.

2.7.5 Test Session Timeout

This was carried out by logging into an account, copying the session ID and Secret Cookie. The browser would then be closed and reopened again, ending the session and deleting the cookies. An attempt was then made to get into the last sessions account by modifying the POST request to include the cookies.

Result:

The session ID and secret cookie both expire at the end of the session in the browser, however the session ID appears to remain for the website. A session ID can be copied before expiring and can be used to replace the new session ID in request headers with OWASP ZAP to regain access to the old session, including access to a customer's account if they have not logged out before closing their session

2.8 DATA VALIDATION TESTING

2.8.1 Testing for Reflected Cross Site Scripting

This was carried out by spidering using OWASP ZAP. It was automatically found.

Result:

A reflected cross site scripting attack is possible in the page "thankyou.php". the URL can be encoded with a script. In the test case, a proof-of-concept alert was executed, however this could be used to send information about users such as a session ID to attackers. The URL used was

<http://192.168.1.20/thankyou.php?id=%3C%2Fh2%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Ch2%3E> and the results can be seen in figure 11.

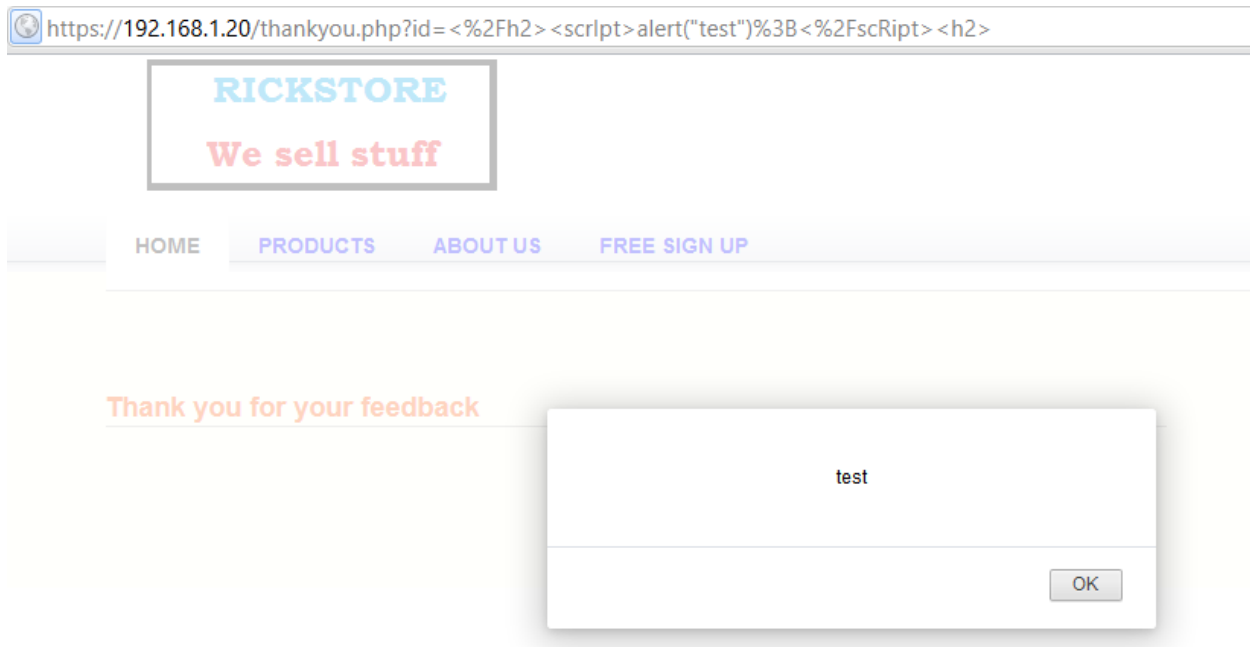


figure 11: successful xss attack

2.8.2 Testing for Stored Cross Site Scripting

OWASP ZAP was initially used to actively scan the website, this found specific web pages vulnerable to persistent cross site scripting. By utilizing the knowledge gained from MySQL testing to create an educated guess of the right format of input. It was known that it uses single quotes so initially, the input `""Fake"><script>alert("test")</script>;"`. This didn't work perfectly, as the last characters were cut when it was attempted. It took several attempts to figure out that certain fields were limited by 25 characters. After this different fields in the employee registration pages were attempted.

Result:

OWASP ZAP found a persistent cross site scripting exploit in the employee registration page. All fields could potentially be used for this exploit. The fields are protected from input by removing non-alphanumeric characters upon entry. This can be bypassed by modifying the POST request in OWASP ZAP to the desired input as the input is not sanitized at that point. The input `""Fake"><script>alert("test")</script>;"` was found to be the correct format to allow scripting. This ended up only working for the image field as all other fields have a max character limit of 25 characters. This was also able to be carried out without needing any account by creating a new customer with the same script in the fields as well as creating a comment in the page `"contact.php"` by inputting the same script in the subject field.

These scripts are easily visible in the admin pages and would need to be obfuscated in order to be effective.

2.8.3 Testing for SQL Injection

This was tested using SQLMap to find potential areas vulnerable to SQL injection. This was essentially an automatic scan without much human intervention and immediately led to a working SQL injection

exploit. The command used was “sqlmap –wizard”, target URL was “192.168.1.20” and everything else was set to default.

Result:

SQLMap found that there is a potential SQL injection at “magaca (POST)”. Inspection of Post requests across the website shows “magaca” is the username in the login pages. SQLMap suggests 3 different types of SQL injections are possible: Boolean-based blind, error based and time-based blind. Each of these exploits can be seen in figure 12 Testing reveals that all 3 types of injection work. This can be exploited in 2 different ways.

If the username of a customer is known, the script “magaca=[username]) AND (SELECT 6308 FROM (SELECT(SLEEP(5)))YXQn)-- vwyC&furaha=&loginkeeping=loginkeeping&submit= Login” can be used to replace the POST request using OWASP ZAP. This will give access to that specific customer’s account. If a username is not known, the script “magaca=UdyP') OR (SELECT 6308 FROM (SELECT(SLEEP(5)))YXQn)-- vwyC&furaha=&loginkeeping=loginkeeping&submit= Login” can be used, which will give access to the first account in the database.

It does not appear that this exploit works for employee accounts. Only customer accounts have been accessed using this exploit.

```
sqlmap identified the following injection point(s) with a total of 8786 HTTP(s) requests:
--
Parameter: magaca (POST)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
Payload: magaca=UdyP') OR NOT 5533=5533#furaha=&loginkeeping=loginkeeping&submit= Login

Type: error-based
Title: MySQL > 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: magaca=UdyP') OR (SELECT 2447 FROM(SELECT COUNT(*),CONCAT(0x716a707071,(SELECT (ELT(2447=2447,1))) ,0x71717a6b71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a) -- hR3b6furaha=&loginkeeping=loginkeeping&submit= Login

Type: time-based blind
Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
Payload: magaca=UdyP') AND (SELECT 6308 FROM (SELECT(SLEEP(5)))YXQn)-- vwyC&furaha=&loginkeeping=loginkeeping&submit= Login
```

figure 12: SQLMap injection results

2.8.4 MySQL Testing

The results of SQLMap are readable in its log file, within it information can be found about the database being used

Result:

SQLMap results showed that the database being used by the server was a MariaDB version of MySQL. It was running version 10.1.31 and used single quotes. An information schema was found which showed that user “root@127.0.0.1” has file privileges

2.9 ERROR HANDLING

2.9.1 Analysis of Error Codes

This testing was carried out throughout the process, any noticeable errors were documented and later reviewed.

Result:

Results of this testing was limited as few errors were found which were at all useful. Errors are present

on all pages most of the time showing that session information was held in the file “/opt/lampp/htdocs/studentsite/usersession.php”. this can be seen in figure 13. Error 404s also show the server banner which displays the information found during information gathering.

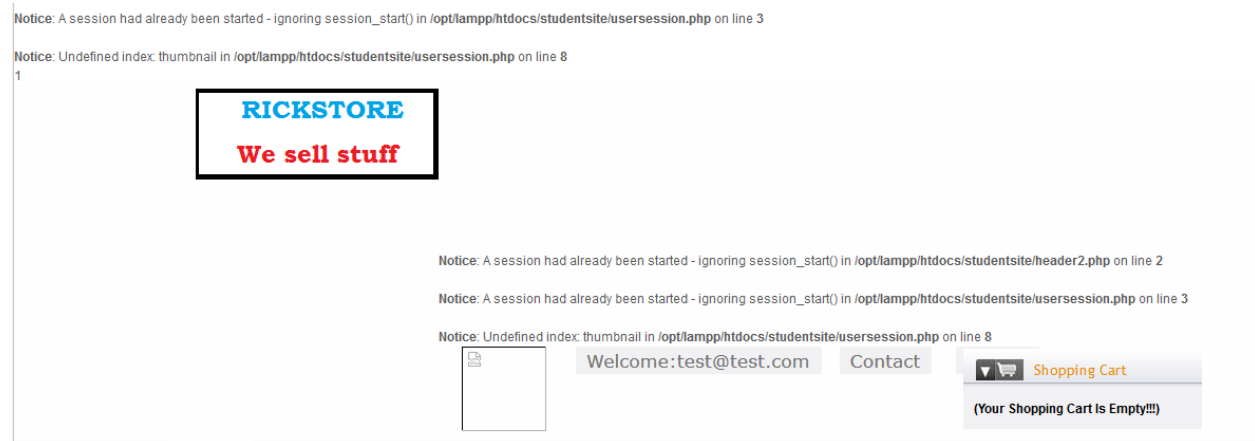


figure 13: Errors on index page

2.10 CRYPTOGRAPHY

2.10.1 Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection

This was carried out using 2 commands. “nmap --script ssl-cert,ssl-enum-ciphers -p 443 192.168.1.20” and “sslyze 192.168.1.20”

Result:

Testing of the SSL ciphers showed several problems. It uses 1024-bit RSA encryption. This is a strong length; however, it has been out of date since 2010. All ciphers used by the website have failed nmap testing. The 64-bit block cipher being used is vulnerable to a SWEET32 attack. It also appears to be using a broken RC4 cipher. MD5 is also being used as the signature certificate which is insecure. Sslyze also believes the SSL certificate is not trusted. Results of these tests can be found in appendices 12 & 13.

2.11 BUSINESS LOGIC TESTING

2.11.1 Test Upload of Unexpected File Types

This was tested by attempting to enter non-image filetypes into 2 image upload inputs in the website, 1 using a customer account and 1 using an employee account. The file used was a python file from a separate assignment.

Result:

It was found that there is some amount of file type validation in use on some parts of the website. Customers can not upload files which aren't images when changing their profile photo. This validation is not present in employee pages such as employee registration, where it was possible to upload a python

file in the employee image field. This means it is also likely that there is no validation for malicious file uploads, however it is unknown for certain as it was not tested.

2.12 CLIENT-SIDE TESTING

2.12.1 Testing for Clickjacking

This was tested using OWASP ZAP to actively scan the website to find pages with missing anti-clickjacking headers

Result:

OWASP ZAP found several pages in the website which were missing anti-clickjacking headers, this means it is potentially vulnerable to clickjacking attacks, however, this was not possible to test any further due to a lack of expertise and the only known tool, “clickjacker tool” appearing to require internet access

2.12.2 Testing WebSockets

This was tested using Google chrome. Inspecting a web page and selecting the “network tab allows current web sockets to be viewed. This was also tested using OWASP ZAP as a proxy for Mantra which has a built in WebSocket viewer.

Result:

WebSockets were searched for using chrome and OWASP ZAP. None were found therefore this test was not applicable.

3 DISCUSSION

3.1 GENERAL DISCUSSION

In general, this website is incredibly insecure and should not be used at all in its current state. Vulnerabilities have been found which relate to 4 of the OWASP top 10 vulnerabilities. Broken access control can be found in the form of violation of the principle of least privilege, bypassing access control checks by modifying the URL, and force browsing to authenticated pages as an unauthenticated user. Cryptography tests showed that the website failed in all but one measure. This is within vulnerability #2 in the top 10 list and should be considered a very high priority to fix due to its prevalence. There are also several injection attacks present in the website in the form of SQL injection and cross site scripting exploits. Some validation and sanitisation of data is attempted for both of these in the website but the current attempts are insufficient and should be improved before the website goes live. There are also several issues related to #6, "Vulnerable and outdated Components". Every single application and architecture being used by the website is out of date. This comes with several unnecessary risks and is easily fixable. These should all be updated to the newest stable version which is compatible with the website.

It is recommended to develop countermeasures and fix all vulnerabilities described in this paper before the website is accessible via the internet. If this is not followed, severe losses could be caused in the form of data loss or disruption to the service. Several exploits could also lead to data breaches of personal information, leading to a loss of reputation and potentially fines as it would be breaking GDPR.

3.2 COUNTERMEASURES

Several countermeasures could be carried out to make this website more secure. OWASP ZAP features a list of alerts categorised in order of severity (seen in appendix 14). This can be found after carrying out an active scan. It is highly recommended to follow the solutions provided in these alerts as that will help fix a significant number of current problems.

It would also be strongly advised to set up and enforce policies such as password requirements and the policy of least privilege. This will help reduce the number of potentially vulnerable accounts in the system as well as reducing the functionality a vulnerable account will have access to, meaning a breach might not be as impactful.

After applying fixes to the website, it may be advisable to reattempt some tests outlined in this paper to make sure the added defences are still not covering any known exploits or vulnerabilities.

3.3 FUTURE WORK

If given more time, it would be interesting to look for potential vulnerabilities that could be utilised to gain root access to the server hosting the website. This was not possible during the time given to write

this paper as it was outside of the realm of expertise of the author. More time would have been needed to research the vulnerabilities required to do this and the process of carrying out this type of exploit.

Several sub-tests were also not carried out in the latter half of test categories due to a mix of time constraints and lack of knowledge. It would be wise to carry out the remaining tests to find any vulnerabilities that were missed in this first iteration of testing.

REFERENCES

- Accenture, 2019. *The cost of cybercrime*. [online] Accenture, p.17. Available at: <https://www.accenture.com/_acnmedia/PDF-96/Accenture-2019-Cost-of-Cybercrime-Study-Final.pdf> [Accessed 10 May 2022].
- Cvedetails.com. n.d. *PHP PHP : List of security vulnerabilities*. [online] Available at: <https://www.cvedetails.com/vulnerability-list/vendor_id-74/product_id-128/PHP-PHP.html> [Accessed 10 May 2022].
- Cvedetails.com. n.d. *Jquery Jquery : List of security vulnerabilities*. [online] Available at: <https://www.cvedetails.com/vulnerability-list/vendor_id-6538/product_id-11031/Jquery-Jquery.html> [Accessed 10 May 2022].
- https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/05-Authorization_Testing/02-Testing_for_Bypassing_Authorization_Schema
- Grossman, J. 2003. Cross-Site Tracing (XST) – The new techniques and emerging threats to bypass current web security measures using TRACE and XSS. White Hat Security, [online], available at: https://www.cgisecurity.com/whitehat-mirror/WH-WhitePaper_XST_ebook.pdf [Accessed 10 May 2022]

APPENDICES

APPENDIX A

1. HTTPPrint report

web server fingerprinting report						
host	port	ssl	banner reported	banner deduced	icon	confidence
192.168.1.20	80		Apache/2.4.29 (Ubuntu) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/5.16.3	Apache/2.0.x		
SSL analysis						

2. Nmap report

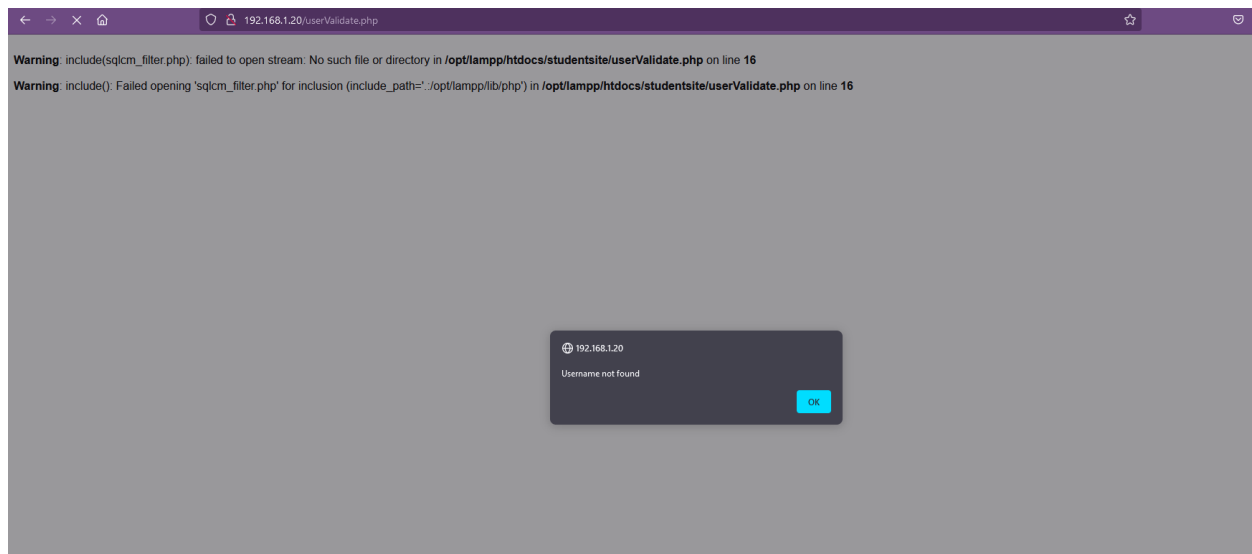
```
root@kali:~# nmap -p 1-65535 -sT 192.168.1.20
Starting Nmap 7.91 ( https://nmap.org ) at 2022-04-27 15:03 EDT
Nmap scan report for 192.168.1.20
Host is up (0.0020s latency).
Not shown: 65531 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http
443/tcp   open  https
3306/tcp  open  mysql
MAC Address: 00:0C:29:E7:76:78 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 2.25 seconds
root@kali:~#
```

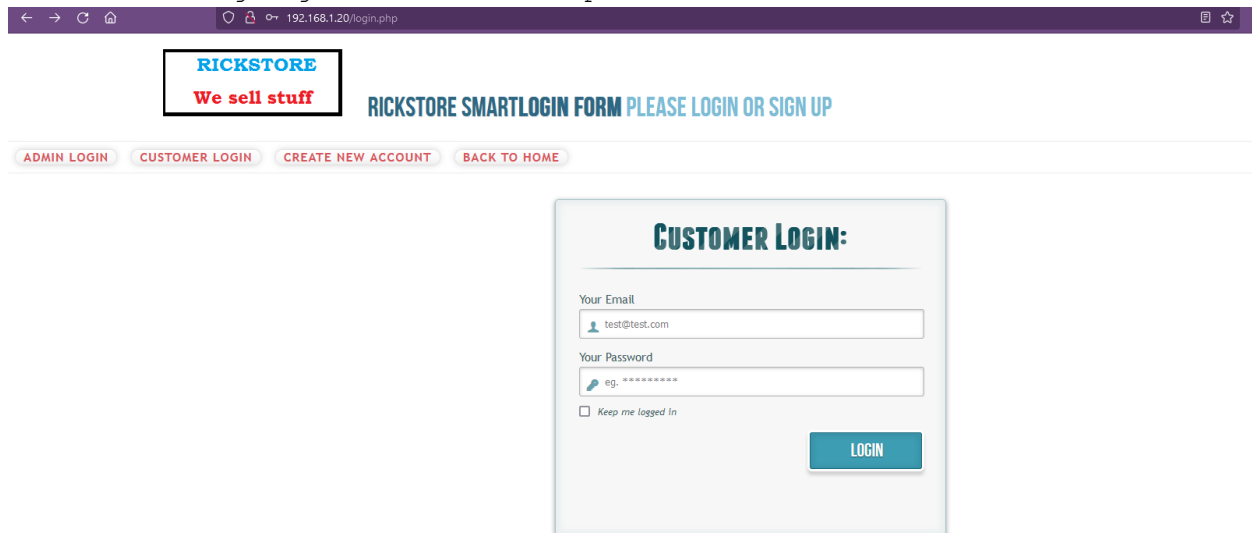
3. Spider results

	A	B	C	D	E
3	TRUE	GET	http://192.168.1.20/robots.txt	Seed	
4	TRUE	GET	http://192.168.1.20/sitemap.xml	Seed	
5	TRUE	GET	http://192.168.1.20/	Seed	
6	TRUE	GET	http://192.168.1.20/admin	Seed	
7	TRUE	GET	http://192.168.1.20/admin/css	Seed	
8	TRUE	GET	http://192.168.1.20/admin/css/bootstrap.min.css	Seed	
9	TRUE	GET	http://192.168.1.20/admin/customerTable.php	Seed	
10	TRUE	GET	http://192.168.1.20/admin/index.php	Seed	
11	TRUE	GET	http://192.168.1.20/admin/shout.php	Seed	
12	TRUE	GET	http://192.168.1.20/css	Seed	
13	TRUE	GET	http://192.168.1.20/css/audioplayer.css	Seed	
14	TRUE	GET	http://192.168.1.20/css/bootstrap.min.css?version=3	Seed	
15	TRUE	GET	http://192.168.1.20/css/cart.css	Seed	
16	TRUE	GET	http://192.168.1.20/css/cart.css?version=1	Seed	
17	TRUE	GET	http://192.168.1.20/css/chatStyle.css	Seed	
18	TRUE	GET	http://192.168.1.20/css/proStyle.css	Seed	
19	TRUE	GET	http://192.168.1.20/css/style.css	Seed	
20	TRUE	GET	http://192.168.1.20/css/style.css?version=17	Seed	
21	TRUE	GET	http://192.168.1.20/css/userlogin.css	Seed	
22	TRUE	GET	http://192.168.1.20/customer.php	Seed	
23	TRUE	GET	http://192.168.1.20/employeeValidate.php	Seed	
24	TRUE	GET	http://192.168.1.20/images	Seed	
25	TRUE	GET	http://192.168.1.20/index.php	Seed	
26	TRUE	GET	http://192.168.1.20/insertCustomer.php	Seed	
27	TRUE	GET	http://192.168.1.20/js	Seed	
28	TRUE	GET	http://192.168.1.20/js/countries.js	Seed	
29	TRUE	GET	http://192.168.1.20/js/cufon-yui.js	Seed	
30	TRUE	GET	http://192.168.1.20/js/functions.js	Seed	
31	TRUE	GET	http://192.168.1.20/js/jquery-1.6.2.min.js	Seed	
32	TRUE	GET	http://192.168.1.20/js/jquery.jcarousel.min.js	Seed	
33	TRUE	GET	http://192.168.1.20/js/main.js	Seed	
34	TRUE	GET	http://192.168.1.20/js/Myriad_Pro_700.font.js	Seed	
35	TRUE	GET	http://192.168.1.20/login.php	Seed	
36	TRUE	GET	http://192.168.1.20/logout.php	Seed	
37	TRUE	GET	http://192.168.1.20/pictures	Seed	
38	TRUE	GET	http://192.168.1.20/pictures/	Seed	
39	TRUE	GET	http://192.168.1.20/products.php	Seed	

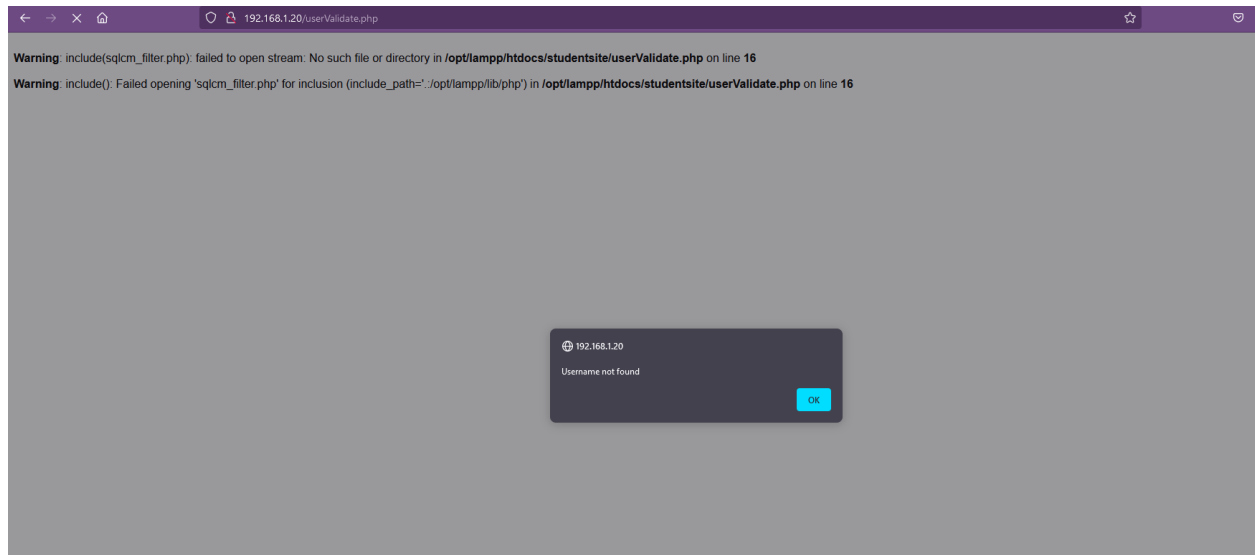
4. Nikto results



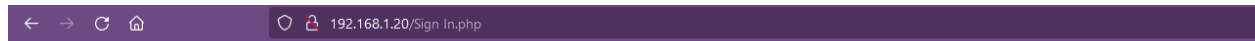
7. Customer login good username bad password



8. Customer login bad username good password



9. Employee login bad username bad password



Object not found!

The requested URL was not found on this server. The link on the [referring page](#) seems to be wrong or outdated. Please inform the author of [that page](#) about the error.

If you think this is a server error, please contact the [webmaster](#).

Error 404

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

10. Employee login good username bad password



Object not found!

The requested URL was not found on this server. The link on the [referring page](#) seems to be wrong or outdated. Please inform the author of [that page](#) about the error.

If you think this is a server error, please contact the [webmaster](#).

Error 404

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

11. Employee login bad username good password

Object not found!

The requested URL was not found on this server. The link on the [referring page](#) seems to be wrong or outdated. Please inform the author of [that page](#) about the error.

If you think this is a server error, please contact the [webmaster](#).

Error 404

[192.168.1.20](#)

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

12. Sslyze results

```

SCAN RESULTS FOR 192.168.1.20:443 - 192.168.1.20
Warning: you are using the root account. You may harm your system.

* TLS 1.1 Cipher Suites:
  Attempted to connect using 80 cipher suites.

  The server accepted the following 17 cipher suites:
    TLS_RSA_WITH_SEED_CBC_SHA          128
    TLS_RSA_WITH_RC4_128_SHA            128
    TLS_RSA_WITH_CAMELLIA_256_CBC_SHA   256
    TLS_RSA_WITH_CAMELLIA_128_CBC_SHA   128
    TLS_RSA_WITH_AES_256_CBC_SHA        256
    TLS_RSA_WITH_AES_128_CBC_SHA        128
    TLS_RSA_WITH_3DES_EDE_CBC_SHA       168
    TLS_ECDHE_RSA_WITH_RC4_128_SHA      128      ECDH: prime256v1 (256 bi
ts)
    TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA   256      ECDH: prime256v1 (256 bi
ts)
    TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA   128      ECDH: prime256v1 (256 bi
ts)
    TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA  168      ECDH: prime256v1 (256 bi
ts)
    TLS_DHE_RSA_WITH_SEED_CBC_SHA        128      DH (1024 bits)
    TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA 256      DH (1024 bits)
    TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA 128      DH (1024 bits)
    TLS_DHE_RSA_WITH_AES_256_CBC_SHA      256      DH (1024 bits)
    TLS_DHE_RSA_WITH_AES_128_CBC_SHA      128      DH (1024 bits)
    TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA     168      DH (1024 bits)

  The group of cipher suites supported by the server has the following properties:
    Forward Secrecy      OK - Supported
    Legacy RC4 Algorithm  INSECURE - Supported

* Downgrade Attacks:
  TLS_FALLBACK_SCSV:    OK - Supported

* Elliptic Curve Key Exchange:
  Supported curves:      prime256v1, secp256k1, secp384r1, secp521r1, sect2
83k1, sect283r1, sect409k1, sect409r1, sect571k1, sect571r1
  Rejected curves:      X25519, X448, prime192v1, secp160k1, secp160r1, se
cp160r2, secp192k1, secp224k1, secp224r1, sect163k1, sect163r1, sect163r2, sect193r1, sect19
3r2, sect233k1, sect233r1, sect239k1

* Session Renegotiation:
  Client Renegotiation DoS Attack:  OK - Not vulnerable
  Secure Renegotiation:             OK - Supported

* Deflate Compression:

```

```
* Deflate Compression: /root/.local/share/sqlmap/output/192.168.1.20/
                        OK - Compression disabled
Warning: You are using the root account. You may harm your system.

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

  The server accepted the following 17 cipher suites:
    TLS_RSA_WITH_SEED_CBC_SHA          128
    TLS_RSA_WITH_RC4_128_SHA            128
    TLS_RSA_WITH_CAMELLIA_256_CBC_SHA   256
    TLS_RSA_WITH_CAMELLIA_128_CBC_SHA   128
    TLS_RSA_WITH_AES_256_CBC_SHA        256
    TLS_RSA_WITH_AES_128_CBC_SHA        128
    TLS_RSA_WITH_3DES_EDE_CBC_SHA       168
    TLS_ECDHE_RSA_WITH_RC4_128_SHA      128      ECDH: prime256v1 (256 bi
ts)
    TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA  256      ECDH: prime256v1 (256 bi
ts)
    TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA  128      ECDH: prime256v1 (256 bi
ts)
    TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA 168      ECDH: prime256v1 (256 bi
ts)
    TLS_DHE_RSA_WITH_SEED_CBC_SHA       128      DH (1024 bits)
    TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA 256      DH (1024 bits)
    TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA 128      DH (1024 bits)
    TLS_DHE_RSA_WITH_AES_256_CBC_SHA     256      DH (1024 bits)
    TLS_DHE_RSA_WITH_AES_128_CBC_SHA     128      DH (1024 bits)
    TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA    168      DH (1024 bits)

  The group of cipher suites supported by the server has the following properties:
    Forward Secrecy      OK - Supported
    Legacy RC4 Algorithm  INSECURE - Supported

* OpenSSL Heartbleed:
                        OK - Not vulnerable to Heartbleed

* Certificates Information:
  Hostname sent for SNI: 192.168.1.20
  Number of certificates detected: 1

  Certificate #0 ( _RSAPublicKey )
    SHA1 Fingerprint: c4c9a1dc528d41ac1988f65db62f9ca922fbe711
    Common Name:      localhost
    Issuer:            localhost
    Serial Number:     0
    Not Before:        2004-10-01
    Not After:         2010-09-30
```

```

Public Key Algorithm: root/local/share/_RSAPublicKey 92.168.1.20/
Signature Algorithm: md5
Key Size: 1024
Exponent: 65537
DNS Subject Alternative Names: []

Certificate #0 - Trust
  Hostname Validation: FAILED - Certificate does NOT match server hostnam
  Android CA Store (9.0.0_r9): FAILED - Certificate is NOT Trusted: self signed c
  Apple CA Store (iOS 14, iPadOS 14, macOS 11, watchOS 7, and tvOS 14): FAILED - Certifi
  Java CA Store (jdk-13.0.2): FAILED - Certificate is NOT Trusted: self signed c
  Mozilla CA Store (2021-01-24): FAILED - Certificate is NOT Trusted: self signed c
  Windows CA Store (2021-02-08): FAILED - Certificate is NOT Trusted: self signed c
  Symantec 2018 Deprecation: ERROR - Could not build verified chain (certificat
  Received Chain: localhost
  Verified Chain: ERROR - Could not build verified chain (certificat
  Received Chain Contains Anchor: ERROR - Could not build verified chain (certificat
  Received Chain Order: OK - Order is valid
  Verified Chain contains SHA1: ERROR - Could not build verified chain (certificat

Certificate #0 - Extensions
  OSCP Must-Staple: NOT SUPPORTED - Extension not found
  Certificate Transparency: NOT SUPPORTED - Extension not found

Certificate #0 - OSCP Stapling
  NOT SUPPORTED - Server did not send back an OSCP r
  response

* TLS 1.2 Session Resumption Support:
  With Session IDs: OK - Supported (5 successful resumptions out of 5 attempts).
  With TLS Tickets: OK - Supported.

* ROBOT Attack:
  OK - Not vulnerable.

* SSL 2.0 Cipher Suites:
  Attempted to connect using 7 cipher suites; the server rejected all cipher suites.

```

```

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

The server accepted the following 29 cipher suites:
  TLS_RSA_WITH_SEED_CBC_SHA          128
  TLS_RSA_WITH_RC4_128_SHA           128
  TLS_RSA_WITH_CAMELLIA_256_CBC_SHA  256
  TLS_RSA_WITH_CAMELLIA_128_CBC_SHA  128
  TLS_RSA_WITH_AES_256_GCM_SHA384    256
  TLS_RSA_WITH_AES_256_CBC_SHA256    256
  TLS_RSA_WITH_AES_256_CBC_SHA       256
  TLS_RSA_WITH_AES_128_GCM_SHA256    128
  TLS_RSA_WITH_AES_128_CBC_SHA256    128
  TLS_RSA_WITH_AES_128_CBC_SHA       128
  TLS_RSA_WITH_3DES_EDE_CBC_SHA      168
  TLS_ECDHE_RSA_WITH_RC4_128_SHA     128      ECDH: prime256v1 (256 bi
ts)
  TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 256      ECDH: prime256v1 (256 bi
ts)
  TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 256      ECDH: prime256v1 (256 bi
ts)
  TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA    256      ECDH: prime256v1 (256 bi
ts)
  TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 128      ECDH: prime256v1 (256 bi
ts)
  TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 128      ECDH: prime256v1 (256 bi
ts)
  TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA    128      ECDH: prime256v1 (256 bi
ts)
  TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA   168      ECDH: prime256v1 (256 bi
ts)
  TLS_DHE_RSA_WITH_SEED_CBC_SHA        128      DH (1024 bits)
  TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA 256      DH (1024 bits)
  TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA 128      DH (1024 bits)
  TLS_DHE_RSA_WITH_AES_256_GCM_SHA384   256      DH (1024 bits)
  TLS_DHE_RSA_WITH_AES_256_CBC_SHA256   256      DH (1024 bits)
  TLS_DHE_RSA_WITH_AES_256_CBC_SHA      256      DH (1024 bits)
  TLS_DHE_RSA_WITH_AES_128_GCM_SHA256   128      DH (1024 bits)
  TLS_DHE_RSA_WITH_AES_128_CBC_SHA256   128      DH (1024 bits)
  TLS_DHE_RSA_WITH_AES_128_CBC_SHA      128      DH (1024 bits)
  TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA     168      DH (1024 bits)

The group of cipher suites supported by the server has the following properties:
  Forward Secrecy      OK - Supported
  Legacy RC4 Algorithm  INSECURE - Supported

* TLS 1.3 Cipher Suites:
  Attempted to connect using 5 cipher suites; the server rejected all cipher suites.

* TLS 1.3 Cipher Suites:
  Attempted to connect using 5 cipher suites; the server rejected all cipher suites.

* OpenSSL CCS Injection:
  OK - Not vulnerable to OpenSSL CCS injection

* SSL 3.0 Cipher Suites:
  Attempted to connect using 80 cipher suites; the server rejected all cipher suites.

SCAN COMPLETED IN 5.54 S

```

13. Nmap cipher test

```
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA - F
TLS_RSA_WITH_RC4_128_SHA - F
TLS_RSA_WITH_SEED_CBC_SHA - F
compressors:
  NULL
cipher preference: client
warnings:
  64-bit block cipher 3DES vulnerable to SWEET32 attack
  Broken cipher RC4 is deprecated by RFC 7465
  Insecure certificate signature: MD5
TLSv1.2:
  ciphers:
    TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (dh 1024) - F
    TLS_DHE_RSA_WITH_AES_128_CBC_SHA (dh 1024) - F
    TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (dh 1024) - F
    TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (dh 1024) - F
    TLS_DHE_RSA_WITH_AES_256_CBC_SHA (dh 1024) - F
    TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (dh 1024) - F
    TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (dh 1024) - F
    TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA (dh 1024) - F
    TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (dh 1024) - F
    TLS_DHE_RSA_WITH_SEED_CBC_SHA (dh 1024) - F
    TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (secp256r1) - F
    TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - F
    TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (secp256r1) - F
    TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (secp256r1) - F
    TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - F
    TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (secp256r1) - F
    TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (secp256r1) - F
    TLS_ECDHE_RSA_WITH_RC4_128_SHA (secp256r1) - F
    TLS_RSA_WITH_3DES_EDE_CBC_SHA - F
    TLS_RSA_WITH_AES_128_CBC_SHA - F
    TLS_RSA_WITH_AES_128_CBC_SHA256 - F
    TLS_RSA_WITH_AES_128_GCM_SHA256 - F
    TLS_RSA_WITH_AES_256_CBC_SHA - F
    TLS_RSA_WITH_AES_256_CBC_SHA256 - F
    TLS_RSA_WITH_AES_256_GCM_SHA384 - F
    TLS_RSA_WITH_CAMELLIA_128_CBC_SHA - F
    TLS_RSA_WITH_CAMELLIA_256_CBC_SHA - F
    TLS_RSA_WITH_RC4_128_SHA - F
    TLS_RSA_WITH_SEED_CBC_SHA - F
  compressors:
    NULL
  cipher preference: client
  warnings:
    64-bit block cipher 3DES vulnerable to SWEET32 attack
    Broken cipher RC4 is deprecated by RFC 7465
    Insecure certificate signature: MD5
least strength: F
```

14. ZAP alerts

- ▼ Alerts (31)
 - > Cross Site Scripting (Persistent) (5)
 - > Cross Site Scripting (Reflected) (24)
 - > Path Traversal
 - > SQL Injection (2)
 - > Absence of Anti-CSRF Tokens (78)
 - > Application Error Disclosure (58)
 - > Buffer Overflow (2)
 - > CSP: Wildcard Directive (2)
 - > CSP: script-src unsafe-inline (2)
 - > CSP: style-src unsafe-inline (2)
 - > Content Security Policy (CSP) Header Not Set (124)
 - > Cross-Domain Misconfiguration
 - > Directory Browsing (9)
 - > Missing Anti-clickjacking Header (92)
 - > Parameter Tampering (19)
 - > Vulnerable JS Library (7)
 - > CSP: Notices (2)
 - > Cookie No HttpOnly Flag (6)
 - > Cookie Without Secure Flag
 - > Cookie without SameSite Attribute (7)
 - > Cross-Domain JavaScript Source File Inclusion (4)
 - > Private IP Disclosure (12)
 - > Server Leaks Information via "X-Powered-By" HTTP Response Header
 - > Timestamp Disclosure - Unix (63)
 - > X-Content-Type-Options Header Missing (313)
 - > **Charset Mismatch**
 - > Content-Type Header Missing (20)
 - > Information Disclosure - Sensitive Information in URL (13)
 - > Information Disclosure - Suspicious Comments (33)
 - > Loosely Scoped Cookie (2)
 - > Re-examine Cache-control Directives