MARTIN MWANGI NJOROGE

mwangimartins650@gmailcom

CS-SA08-24031

OWASP Top 10 - 2021

Here's a link to my finished room,

https://tryhackme.com/r/room/owasptop102021

1.0Introduction

This room focuses on breaking down the top 10 OWASP, their vulnerabilities and ways to counter the vulnerabilities to prevent their exploits. It gives an hand-on practical sessions on how these security risks can be exploited.

1.1Broken access

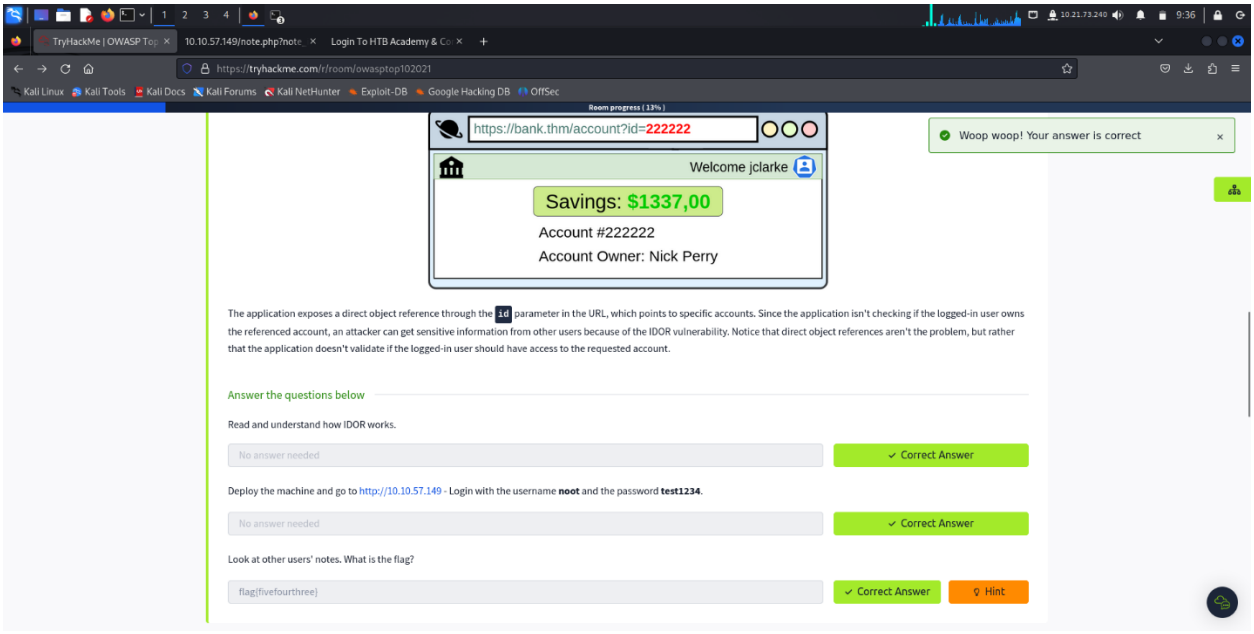
Is a vulnerability in web applications and it can range to many vulnerabilities e.g. bypassing control checks by modifying URLs, internal applications state or html pages. Elevating Privileges, Accessing APIs with missing access controls for PUT, POST or DELETE.

To prevent these vulnerabilities can range to many practices such as Implement strong session management practices to protect user sessions from hijacking, Rate limiting the APIs.

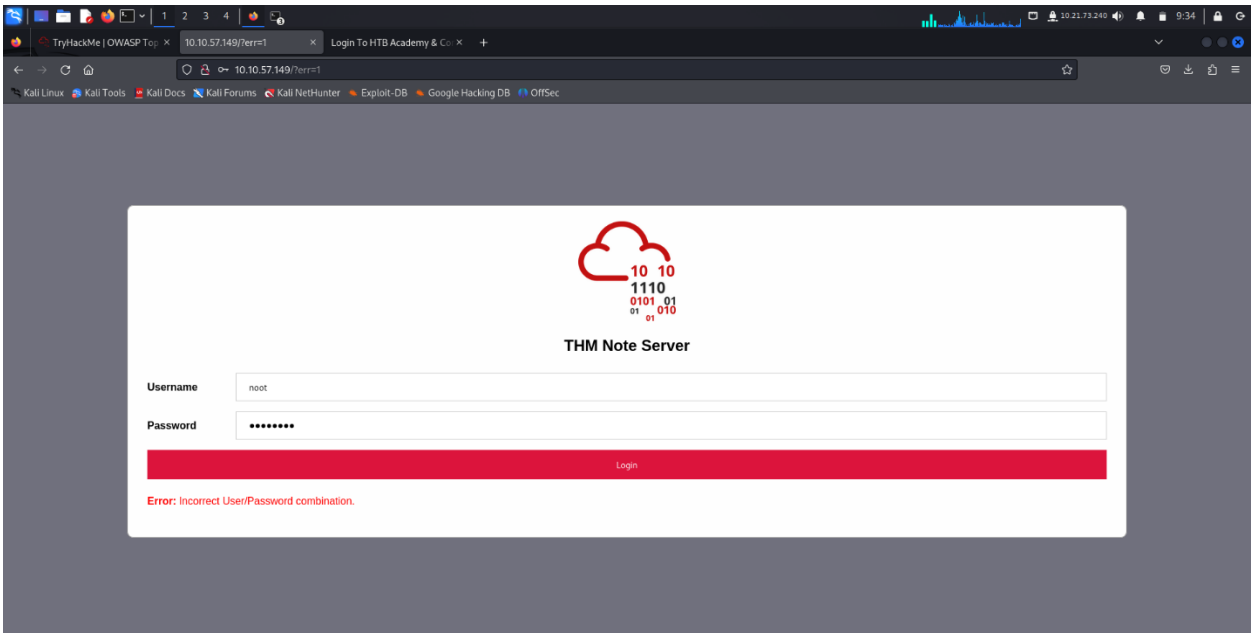
Insecure Direct Object Reference (IDOR) Challenge.

IDOR refers to an access control vulnerability where you can access resources you wouldn't ordinarily be able to see. This occurs when the programmer exposes a Direct Object Reference, which is just an identifier that refers to specific objects within the server.

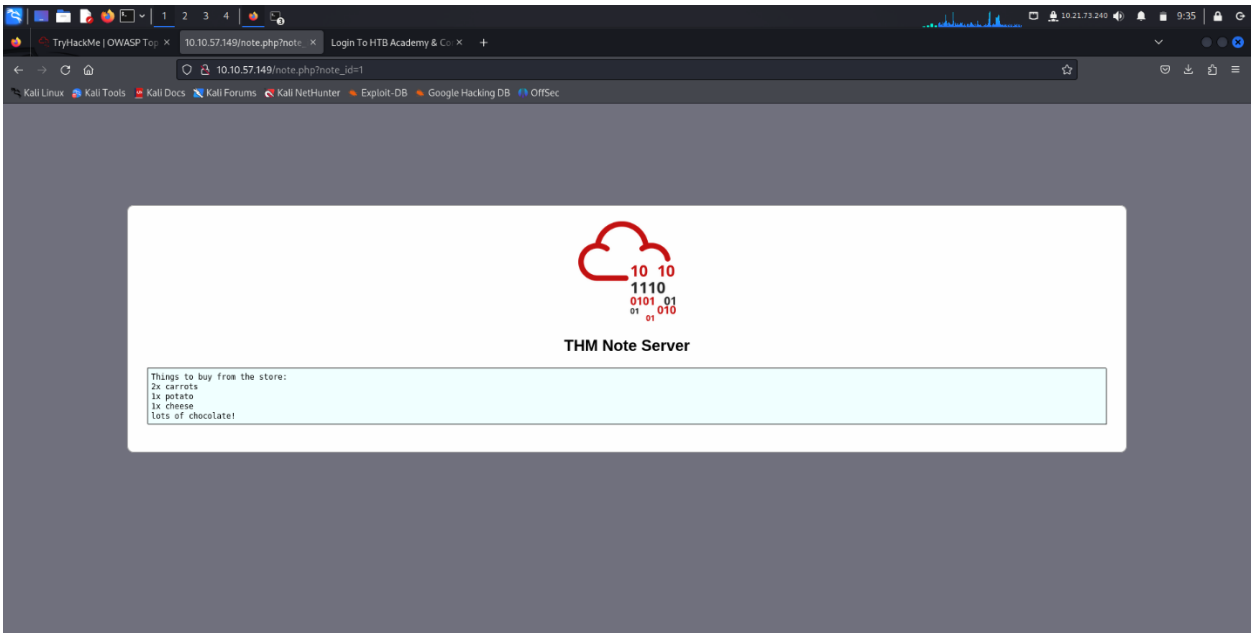
Questions



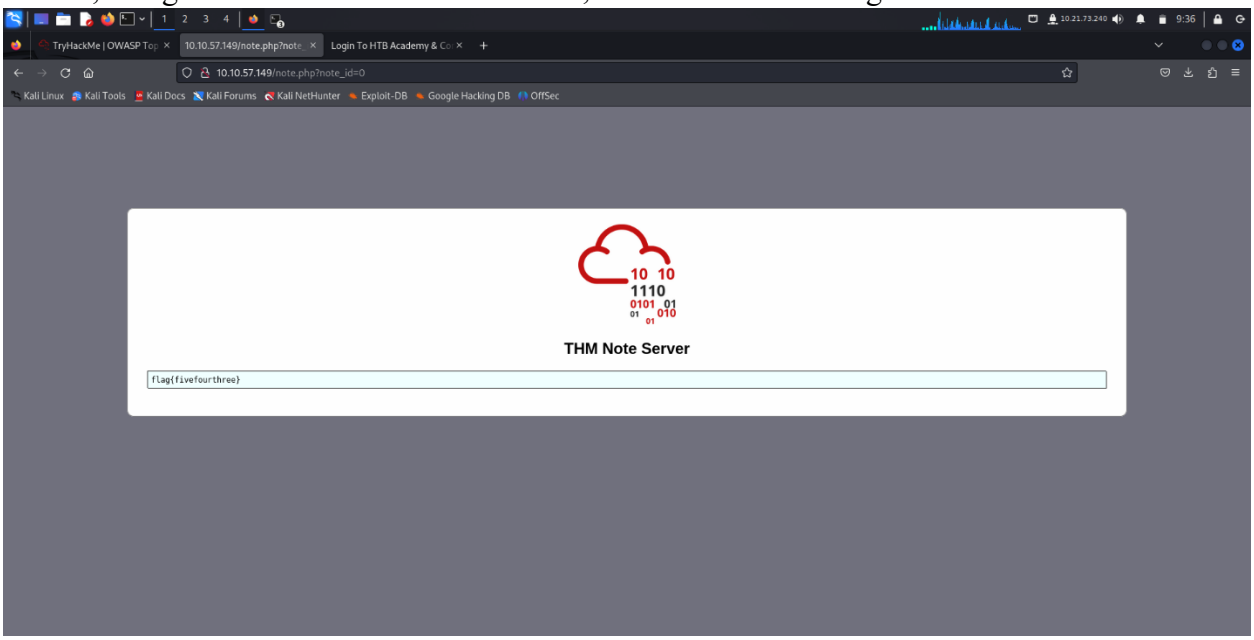
Used the username not and password test1234 to gain access of the site,



After successful login this was the layout of the site and at the URL tab there was an ID parameter that can be manipulated to hijack another users login session,



The site exposes a direct object reference through the id parameter in the URL, it changed the ID no to 0, and go access to another user session, where I found the flag.



1.2Cryptographic Failures

A cryptographic failure refers to any vulnerability arising from the misuse or lack of use of cryptographic algorithms for protecting sensitive information. Web applications require cryptography to provide confidentiality for their users at many levels.

This leads to user data to be in plaintext that can be hijacked and easily be seen, thus involving Man-In-The-Middle attacks.

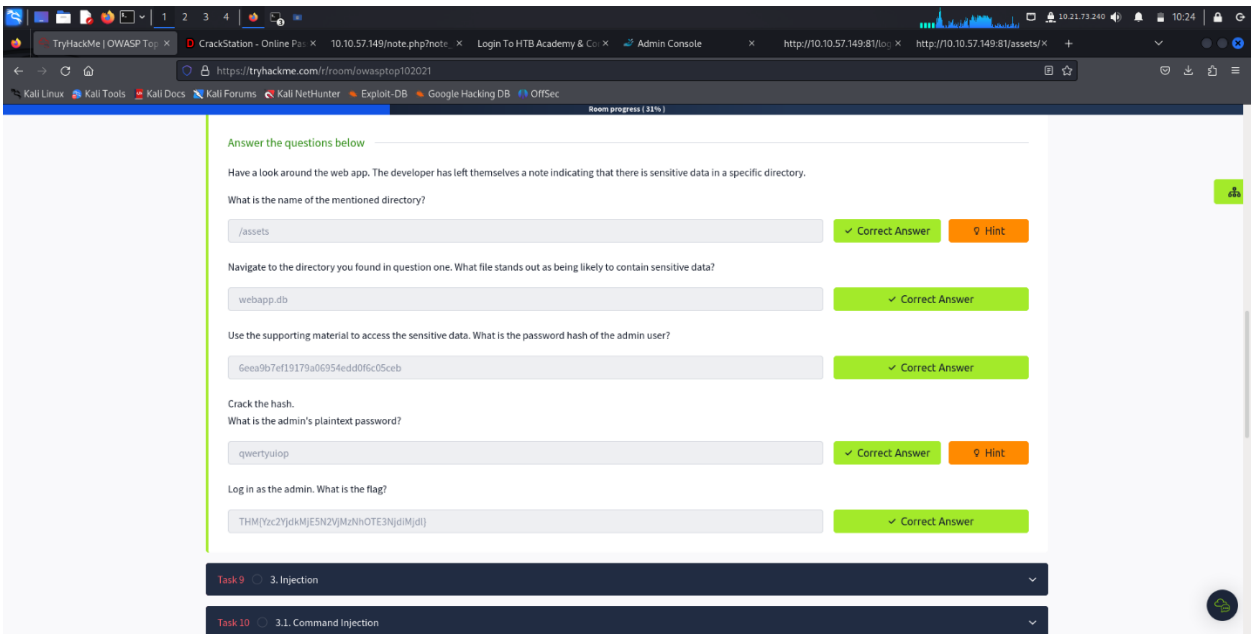
Best way to prevent this is to use the cryptographic algorithms, and encrypting the data during transit or at rest. And avoiding using deprecated cryptographic functions and padding schemes such as MD5, SHA1.

Scenarios

it is common to see databases set up on dedicated servers running a database service such as MySQL or MariaDB; however, databases can also be stored as files. These are referred to as "flat-file" databases, as they are stored as a single file on the computer. This is much easier than setting up an entire database server. If the flat database stored underneath the root directory of the website (i.e. one of the files accessible to the user connecting to the website), we can download and query it on our own machine, with full access to everything in the database.

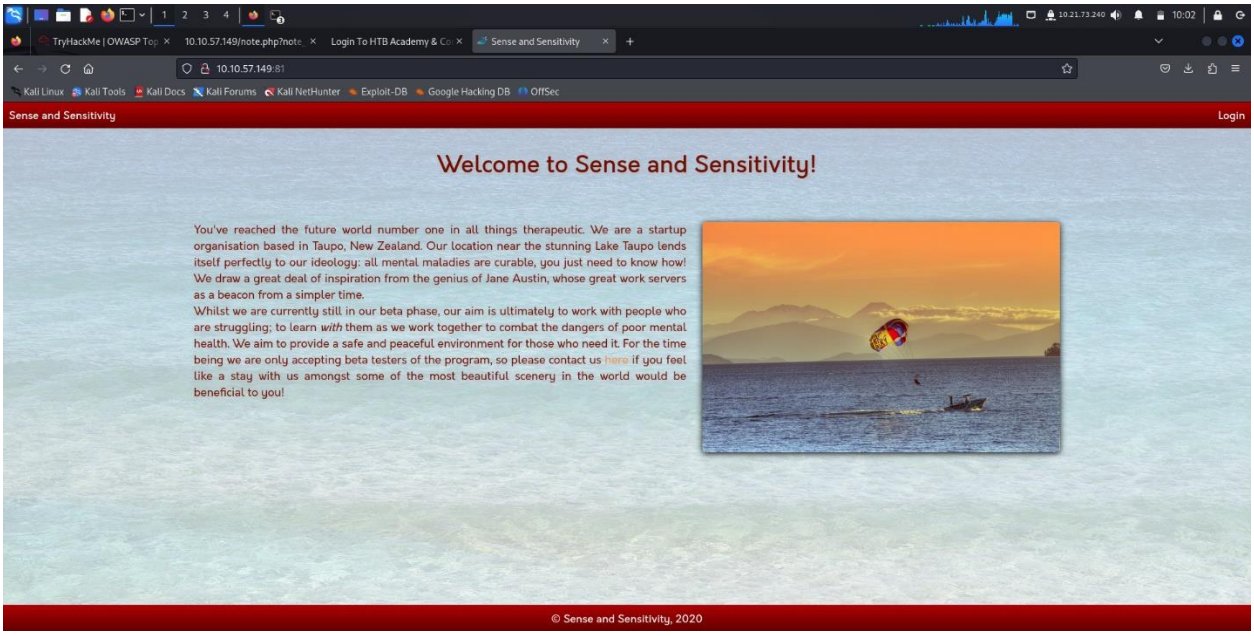
Using a tool such as Sqlite3 we can use it to access the flat-file database. And from the we can use commands like .tables to interact with the database, .tables list any available tables, PRAGMA table_info() list the table information, columns names.

Questions

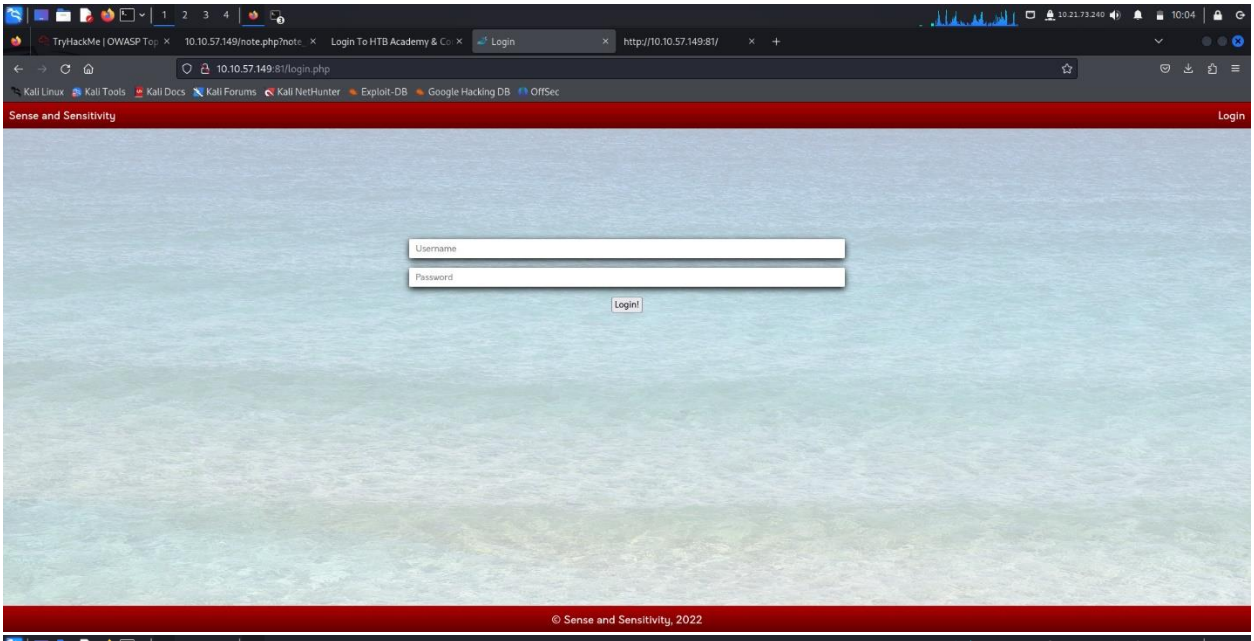


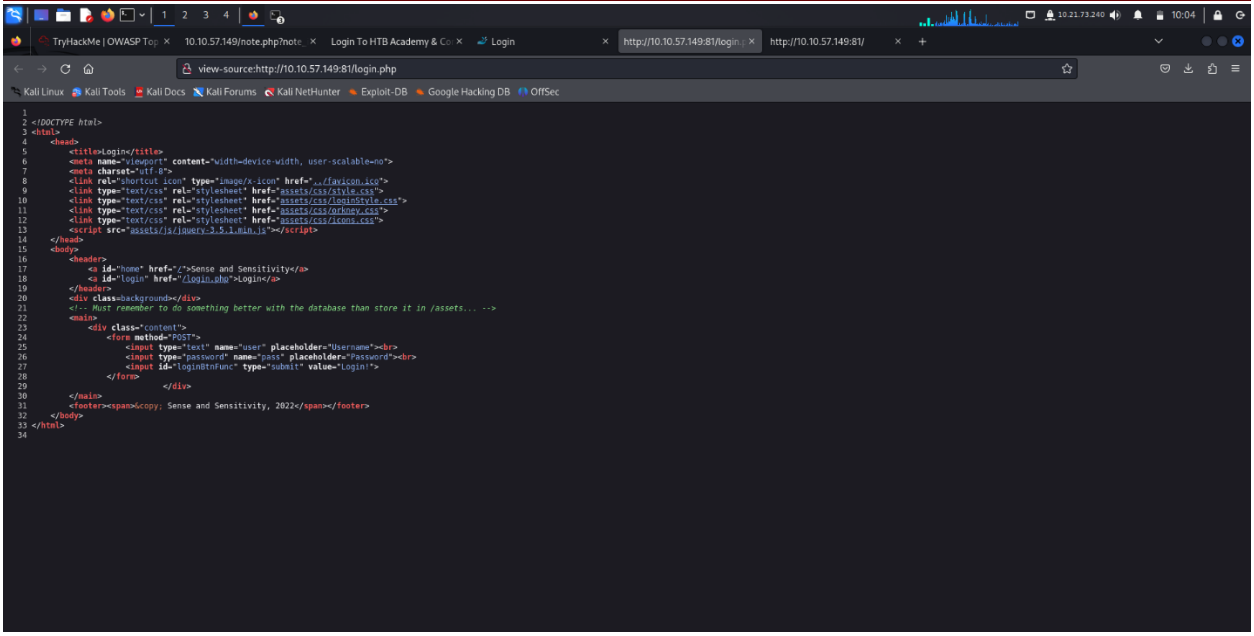
Quiz 1

I first loaded the target IP to see how the site looked like, this was the results,



On the site there was Login pag, which required a username and password, inspecting the login page source code, there was a comment left by the developer about a /assets directory existing.


10.10.57.149:81/login.php
Sense and Sensitivity
Login
Username
Password
Login!
© Sense and Sensitivity, 2022


view-source:http://10.10.57.149:81/login.php
<!DOCTYPE html>
<html>
<head>
<title>Login</title>
<meta name="viewport" content="width=device-width, user-scalable=no">
<meta charset="utf-8">
<link rel="shortcut icon" type="image/x-icon" href="../favicon.ico">
<link type="text/css" rel="stylesheet" href="assets/css/style.css">
<link type="text/css" rel="stylesheet" href="assets/css/loginStyle.css">
<link type="text/css" rel="stylesheet" href="assets/css/orkney.css">
<link type="text/css" rel="stylesheet" href="assets/css/icons.css">
<script src="assets/js/jquery-3.5.1.min.js"></script>
</head>
<body>
<header>
Sense and Sensitivity
Login
</header>

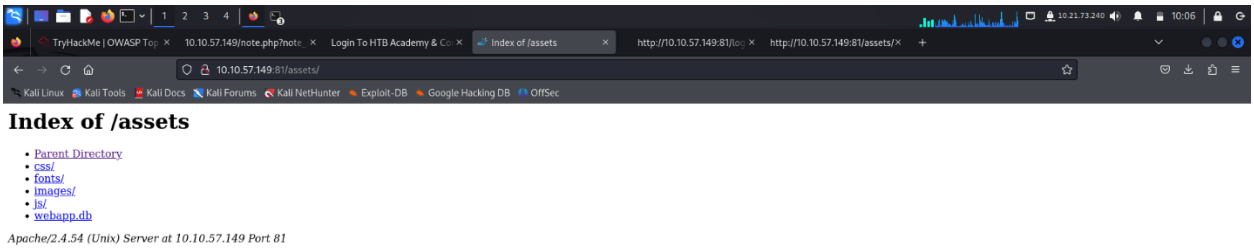
<!-- Must remember to do something better with the database than store it in /assets... -->
<main>

<form method="POST">
<input type="text" name="user" placeholder="Username">
<input type="password" name="pass" placeholder="Password">
<input id="loginBtnFunc" type="submit" value="Login!">
</form>

</main>
<footer>© Sense and Sensitivity, 2022</footer>
</body>
</html>

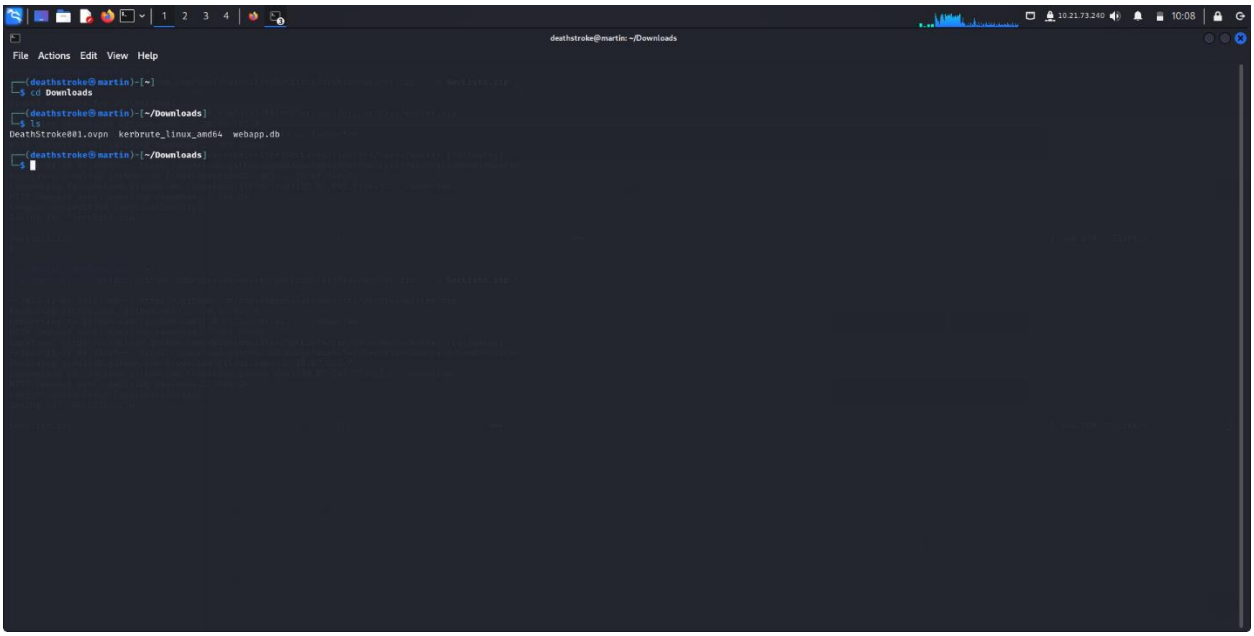
Quiz 2

I investigated the /assets directory by adding the directory name on the URL, I found it had folders and among them there was a database file, webapp.db,

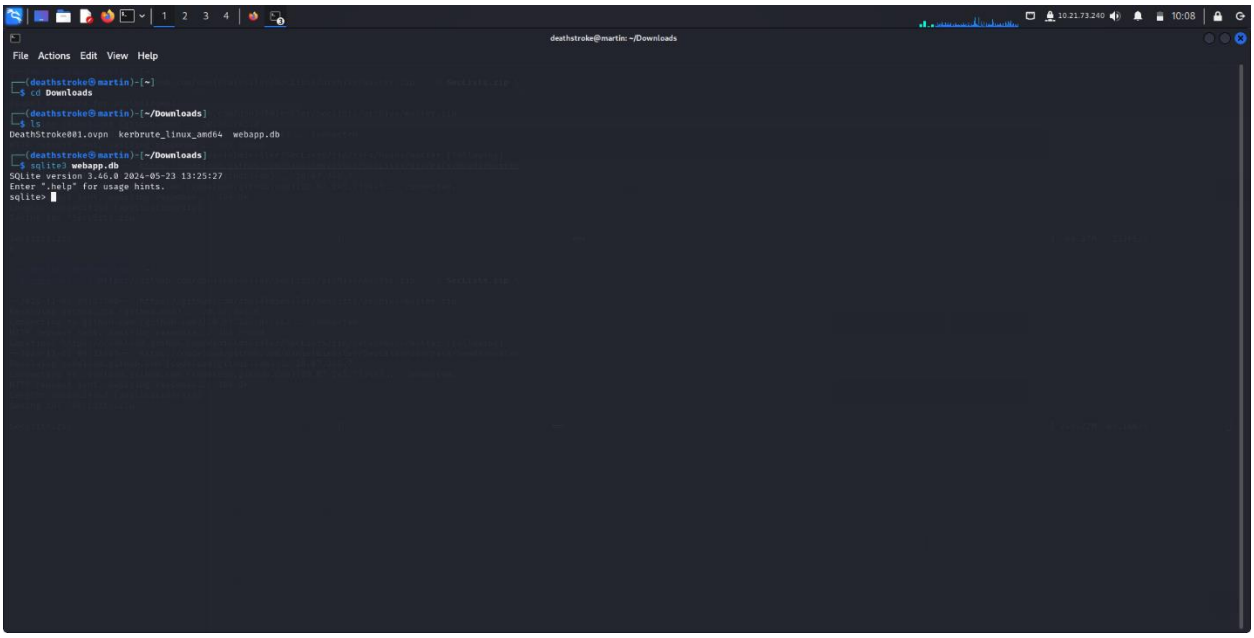


Quiz 3

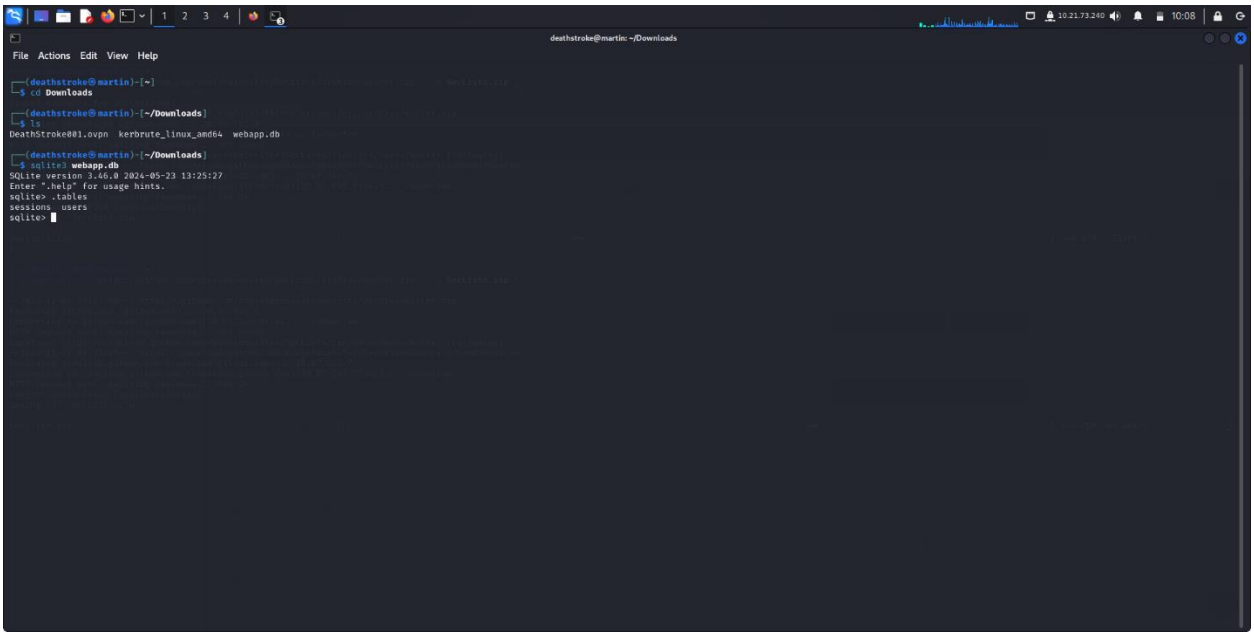
I downloaded the webapp.db file,



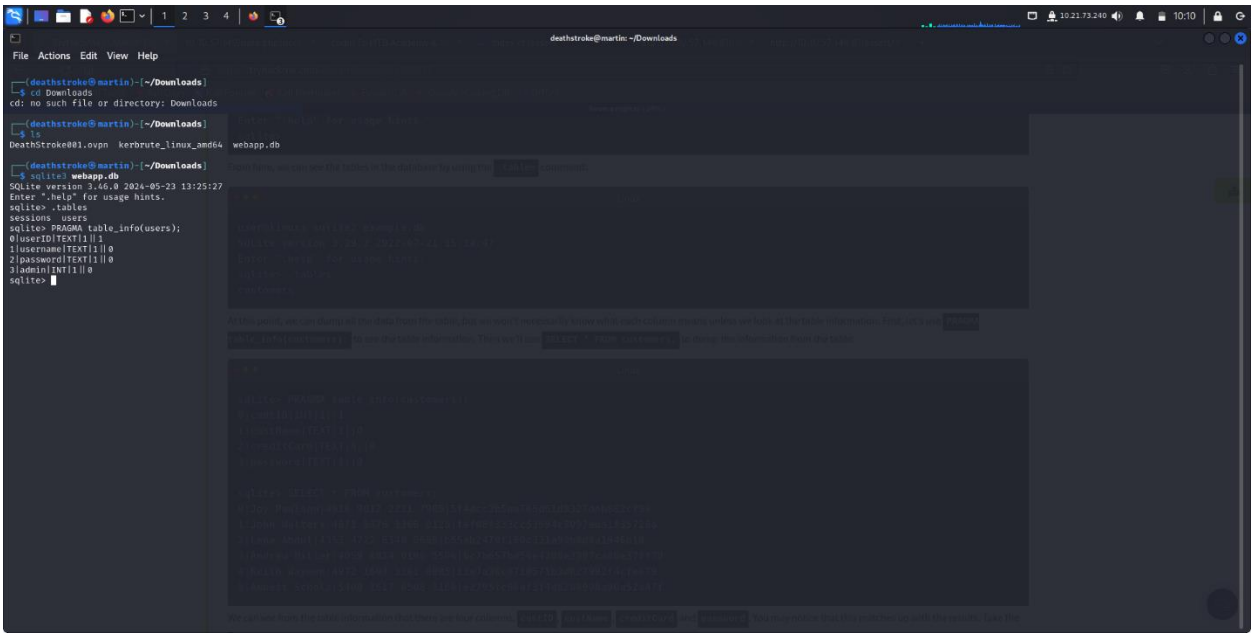
Used *sqlite3* tool to view the details of the database file, the tool created platform to interact with the database,



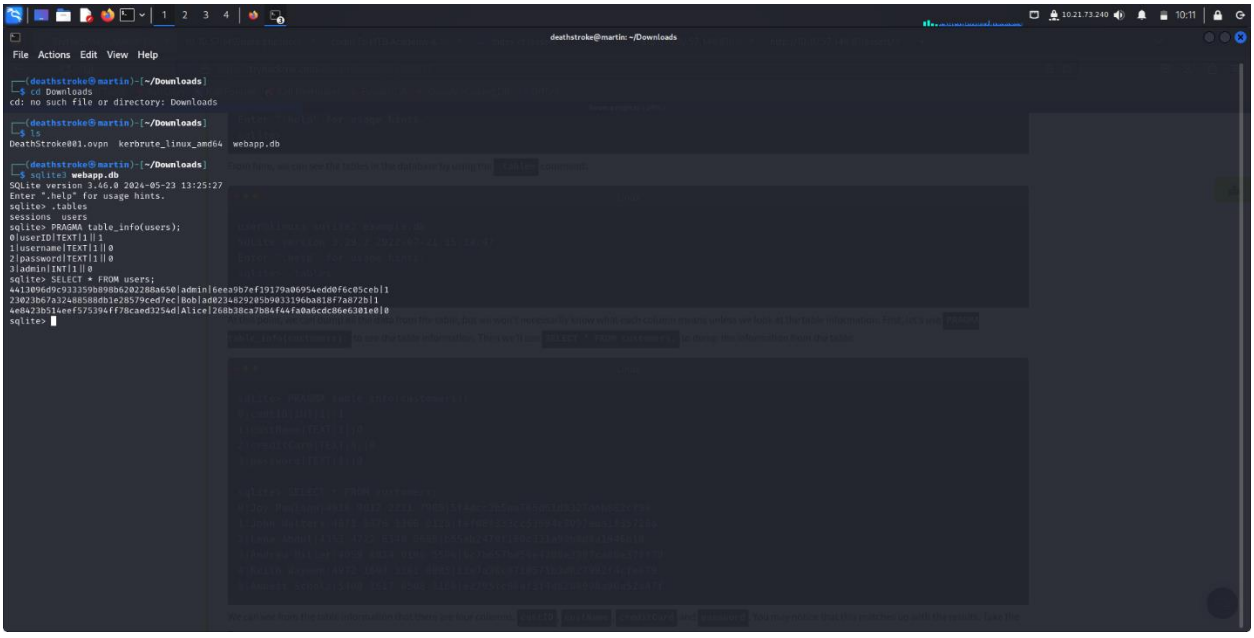
Using the command *.tables* it listed the tables on the database, users, sessions,



After viewing the tables available the users table was the one I required, with use of a command, *PRAGMA table_info();* it listed the contents of the user tables, the username, ID, password and admin

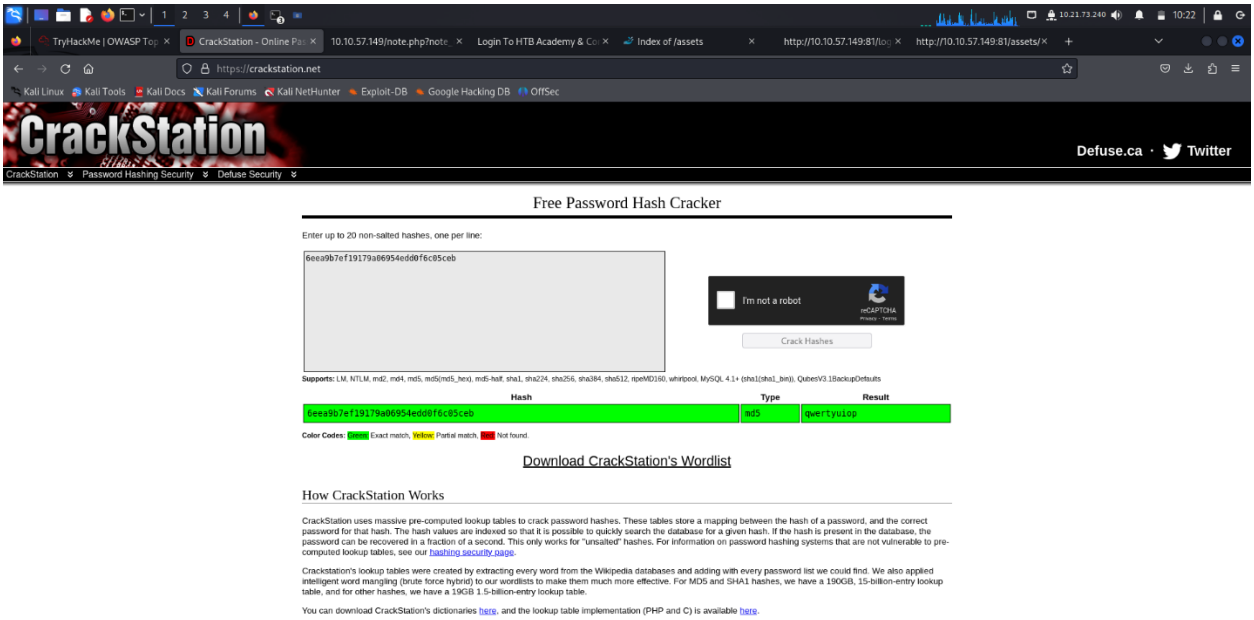


Used sql commands, SELECT all FROM table users, and found the admin user and its password which was hashed,



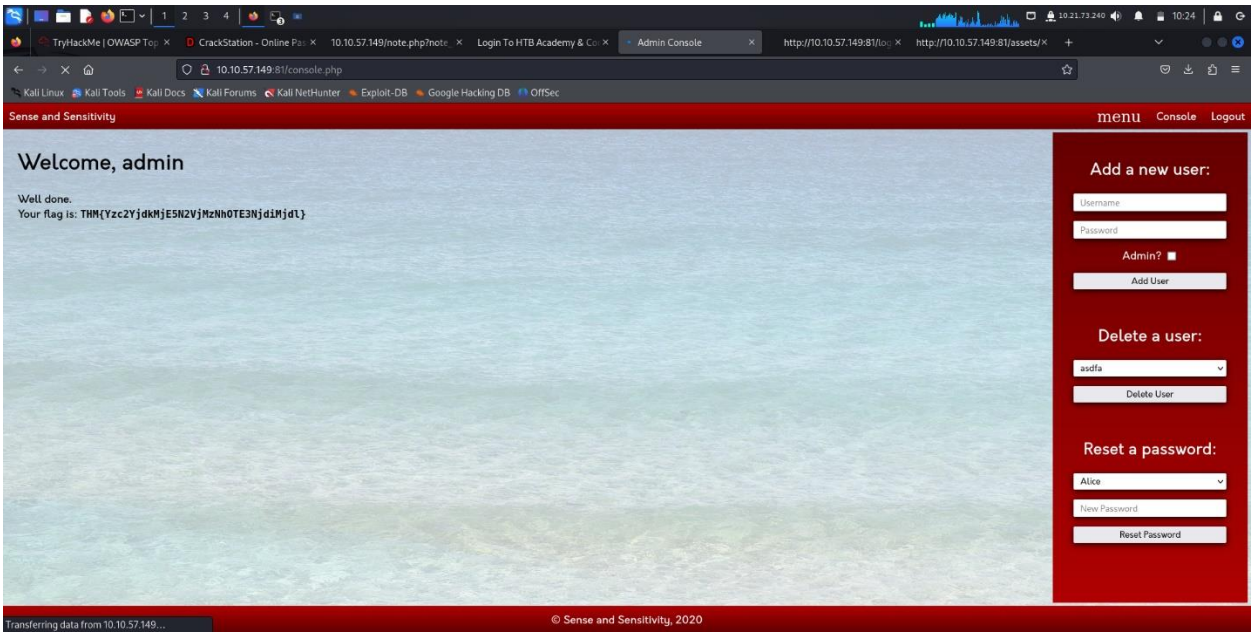
Quiz 4

Used crackstation tool to decrypt the hashed admin password, and this was the results,



Quiz 5

Used the credentials username(admin) and the password I cracked, to login to the admins account it was successful and I found the flag,



1.3Injections

These Injections can be either;

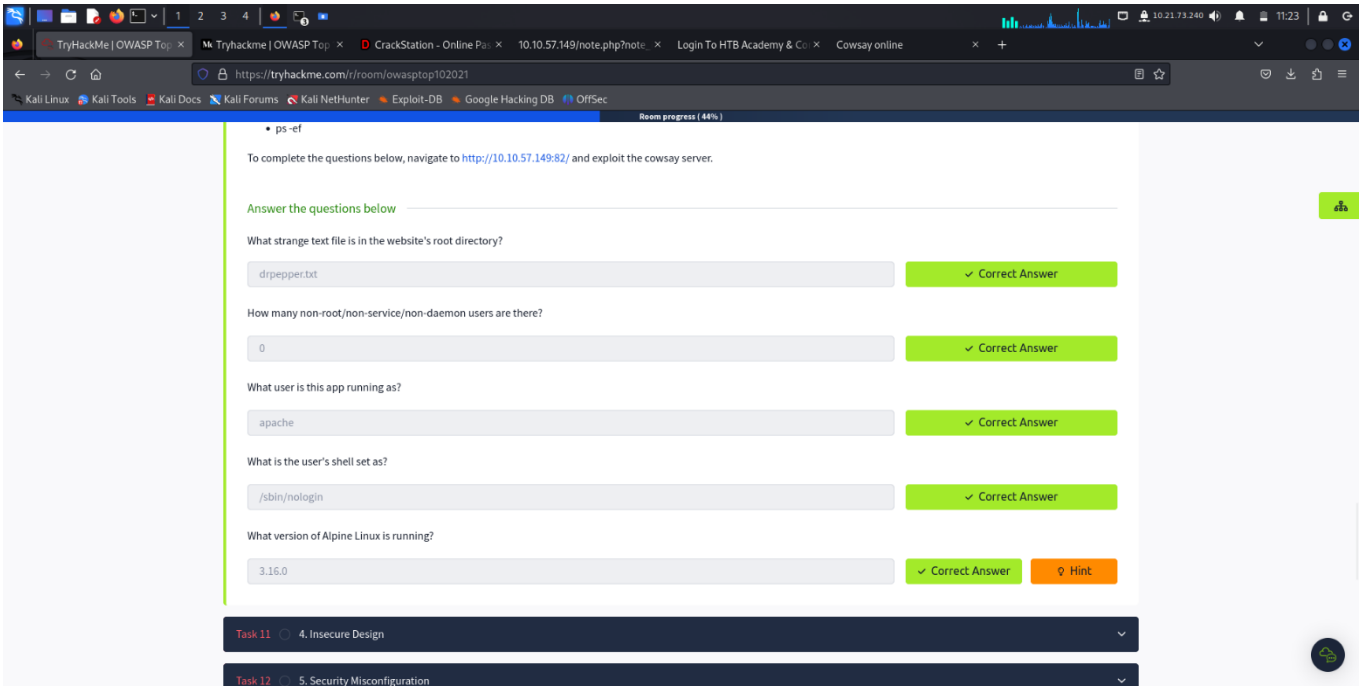
SQL Injection occurs when user-controlled input is passed to SQL queries. As a result, an attacker can pass in SQL queries to manipulate the outcome of such queries. This could potentially allow the attacker to access, modify and delete information in a database when this input is passed into database queries.

Command Injection occurs when user input is passed to system commands. As a result, an attacker can execute arbitrary system commands on application servers, potentially allowing them to access users' systems.

The defense for this is using safe APIs and ensuring that user-controlled input is not interpreted as queries or commands.

Command Injection occurs when server-side code (like PHP) in a web application makes a call to a function that interacts with the server's console directly.

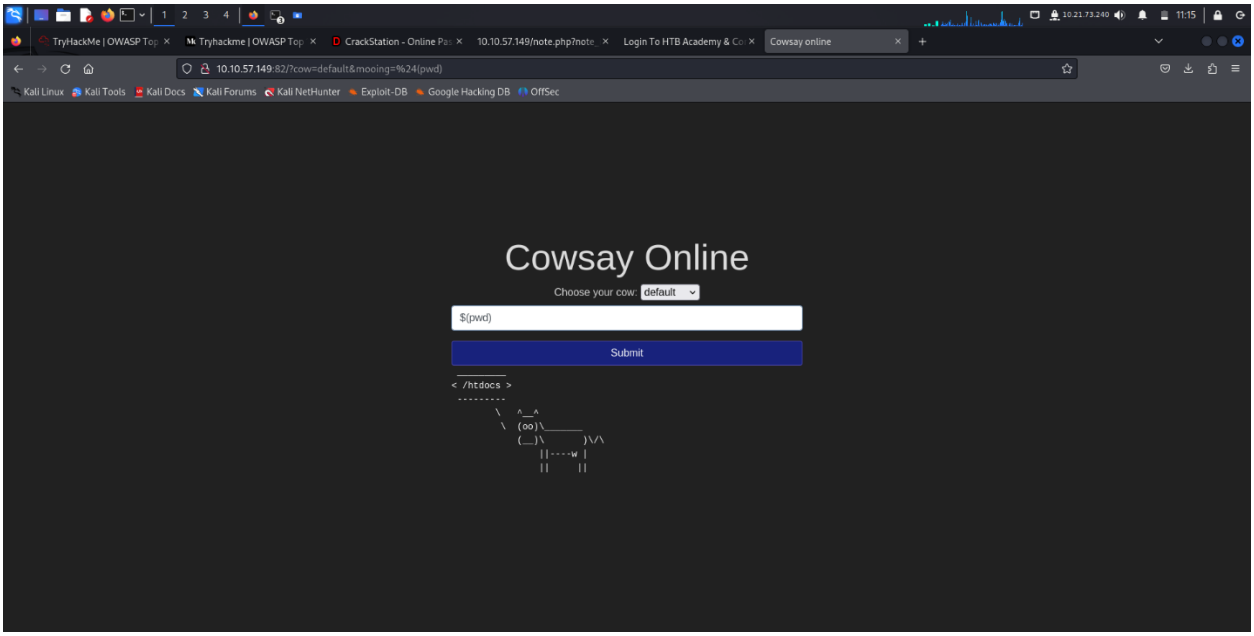
Questions



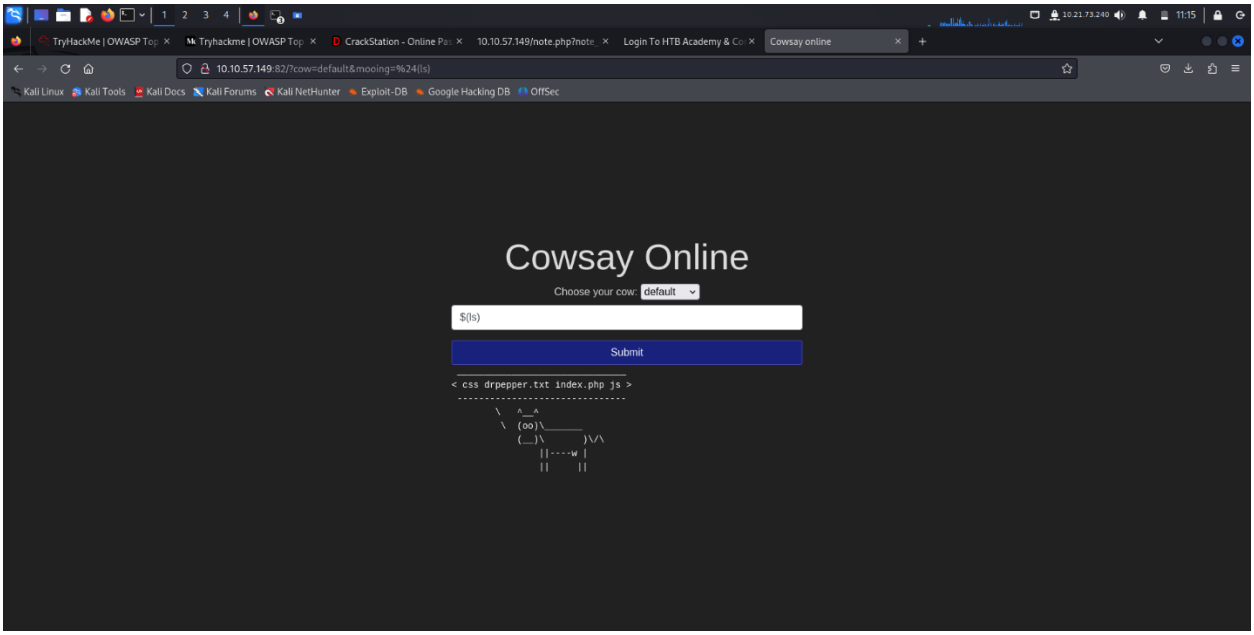
Quiz 1

Using my target IP, loaded it on a firefox, where it gave a platform to choose your cow, with this interface I performed command injection, using inline commands where I wrapped my commands with a dollar sign and brackets, like $(command),

The first command was pwd to print working directory to understand in which directory I was operating at,

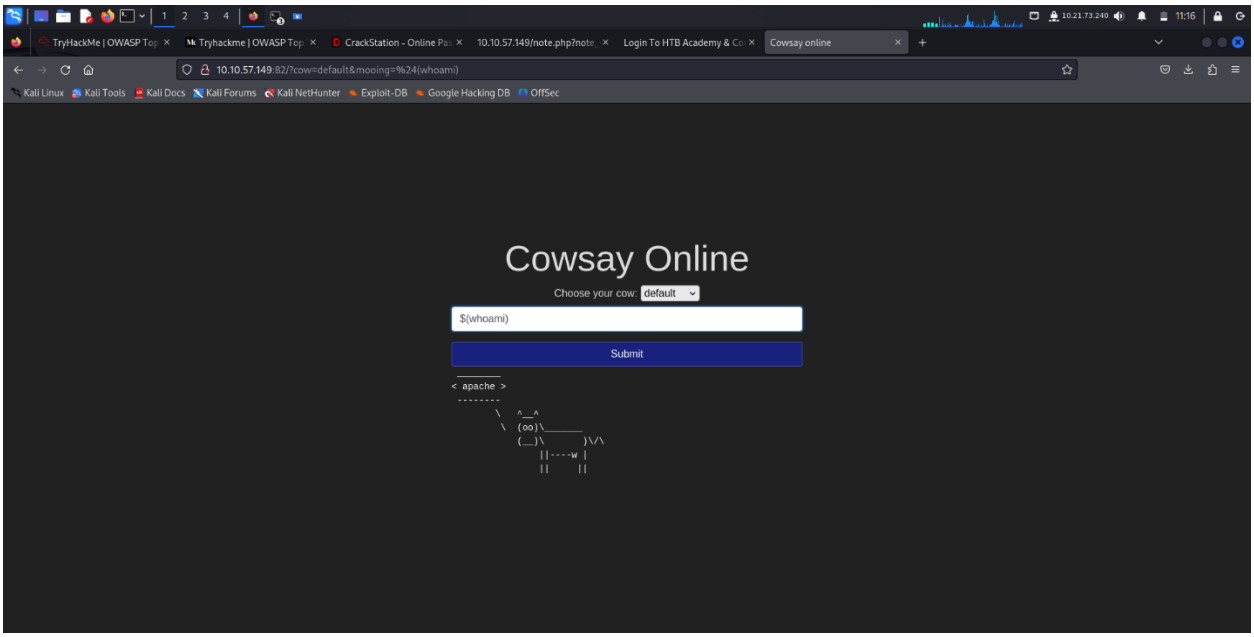


I used command ls to list files within the directory htdocs, and found one as the answer,



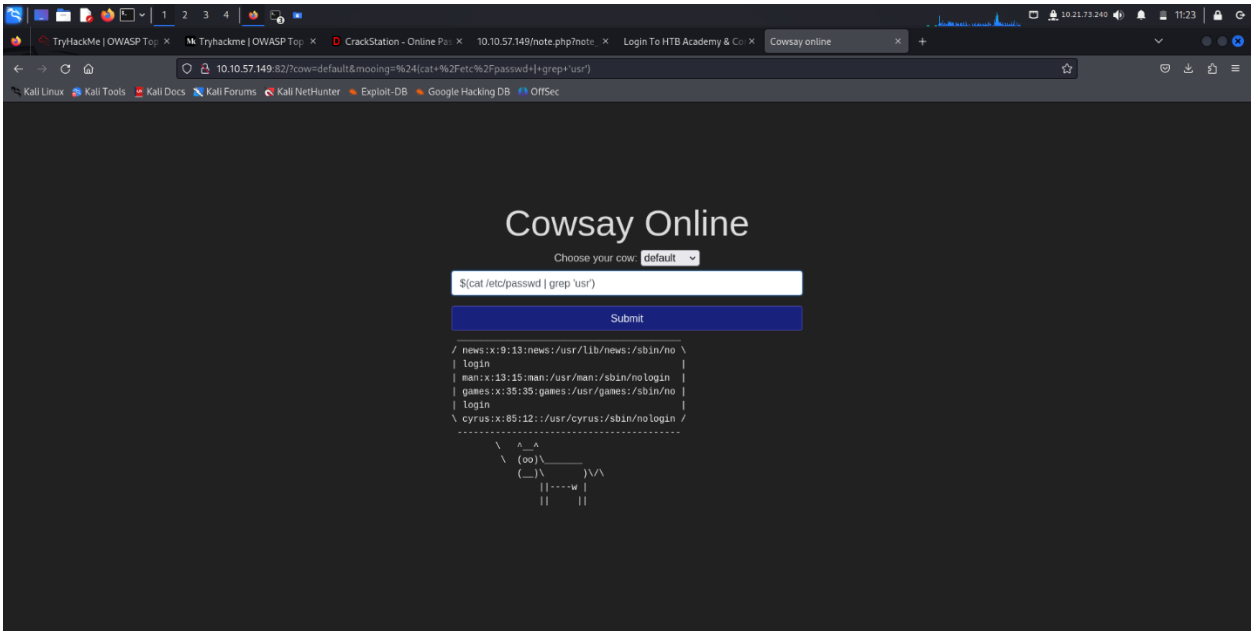
Quiz 3

Used whoami command to view the current username,



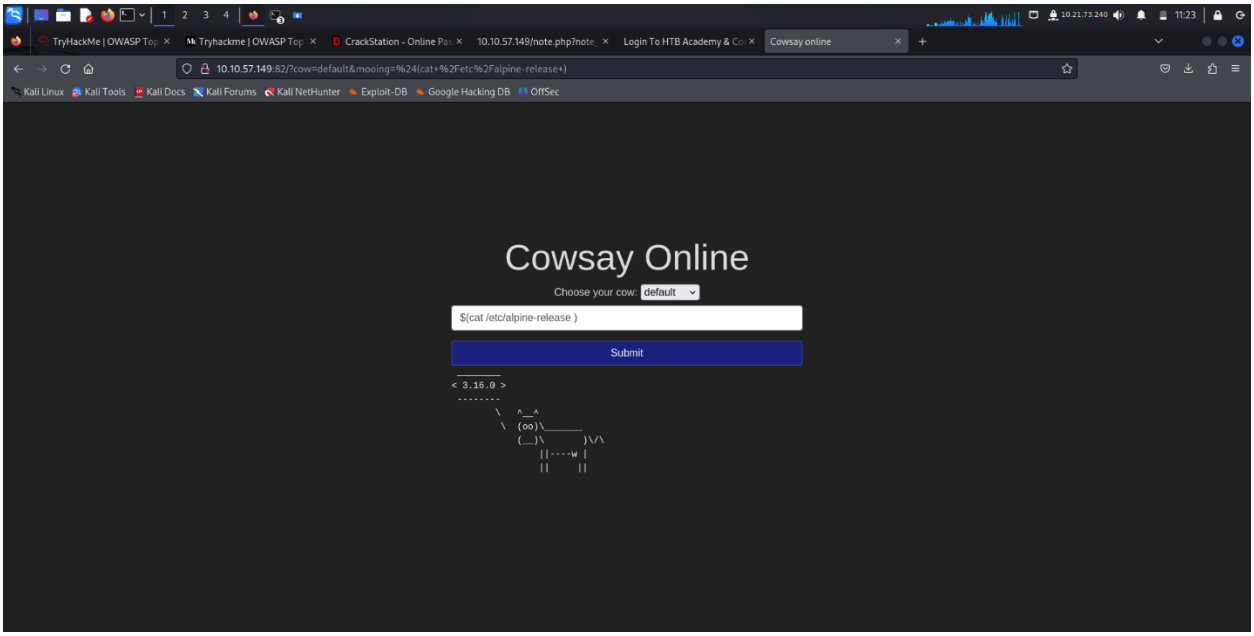
Quiz 4

Used cat command with path to passwd folder and piped command grep to select only usr,



Quiz 5

Used cat command and path to alpine-release to view the version of the alpine,



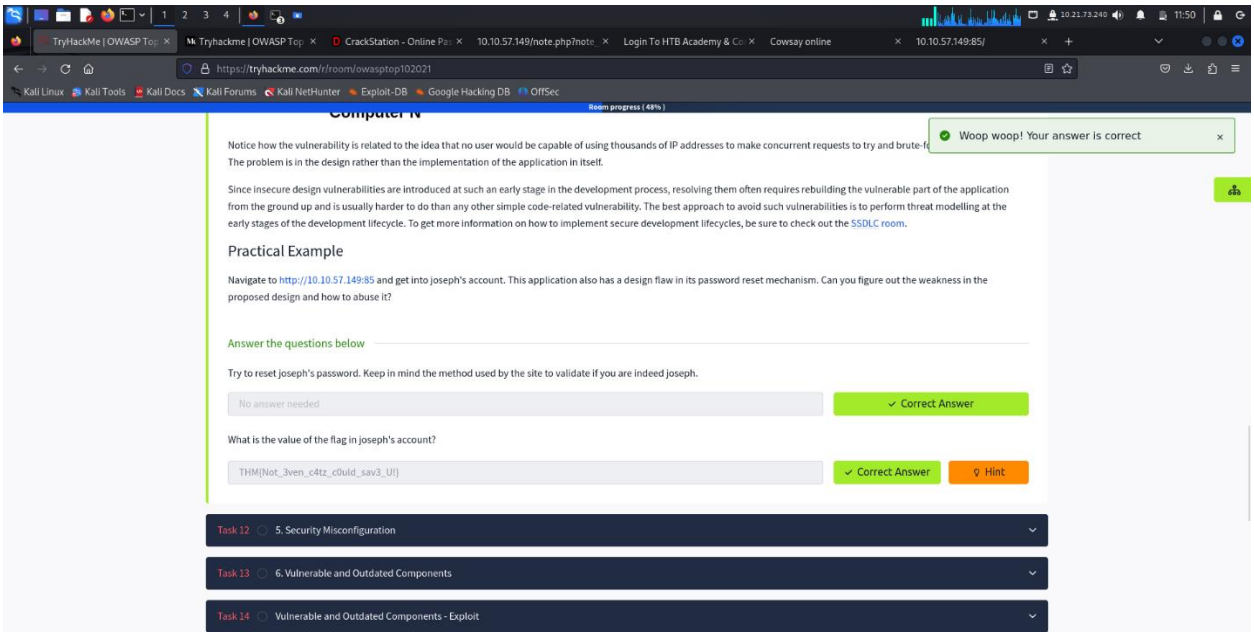
1.4Insecure Design

Insecure design refers to vulnerabilities which are inherent to the application's architecture. They are not vulnerabilities regarding bad implementations or configurations, but the idea behind the whole application is flawed from the start.

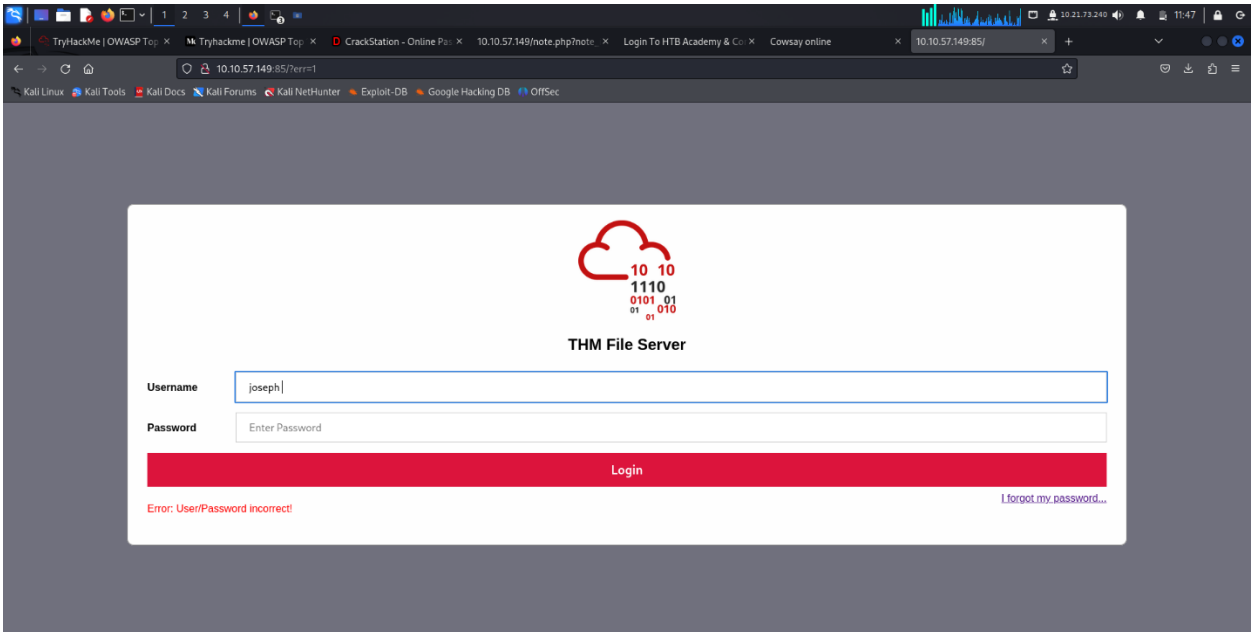
these vulnerabilities occur when an improper threat modelling is made during the planning phases of the application and propagate all the way up to your final app. It may also be introduced by developers while adding some "shortcuts" around the code to make their testing easier.

The best way to prevent this includes performing threat modelling at the early stages of the development lifecycle, establishing and using secure development lifecycle with AppSec professionals to help elevate and design security and privacy controls.

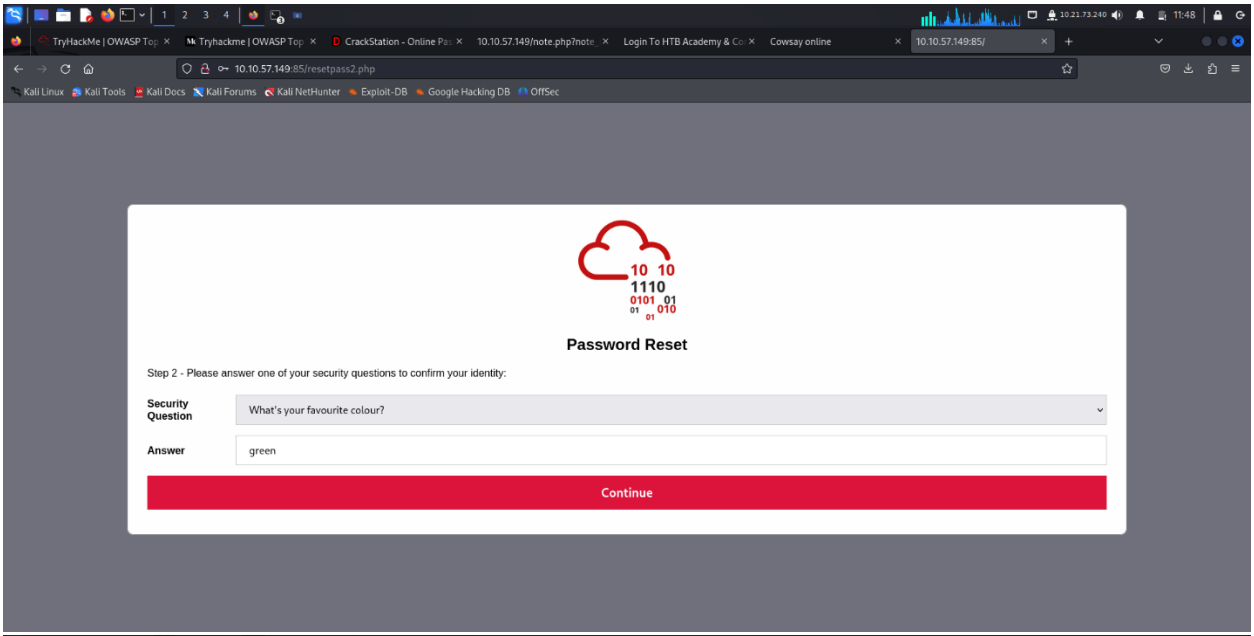
Questions

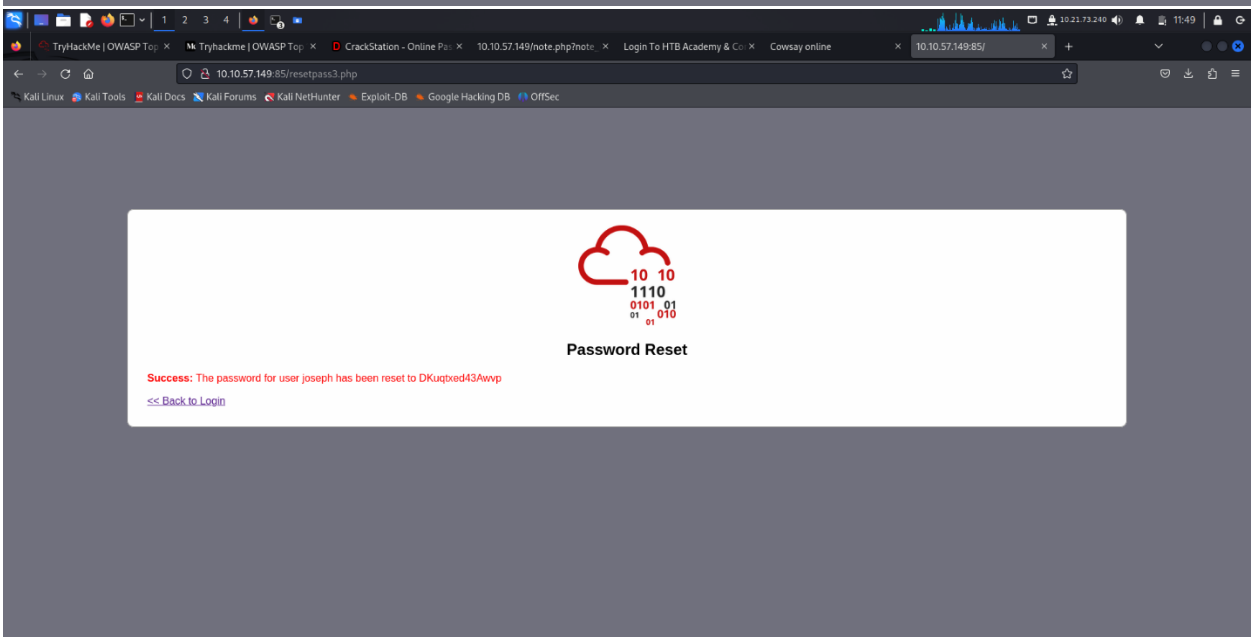


I launched the site using the target IP, and this was the results, it was a login platform, I had the username but no password,

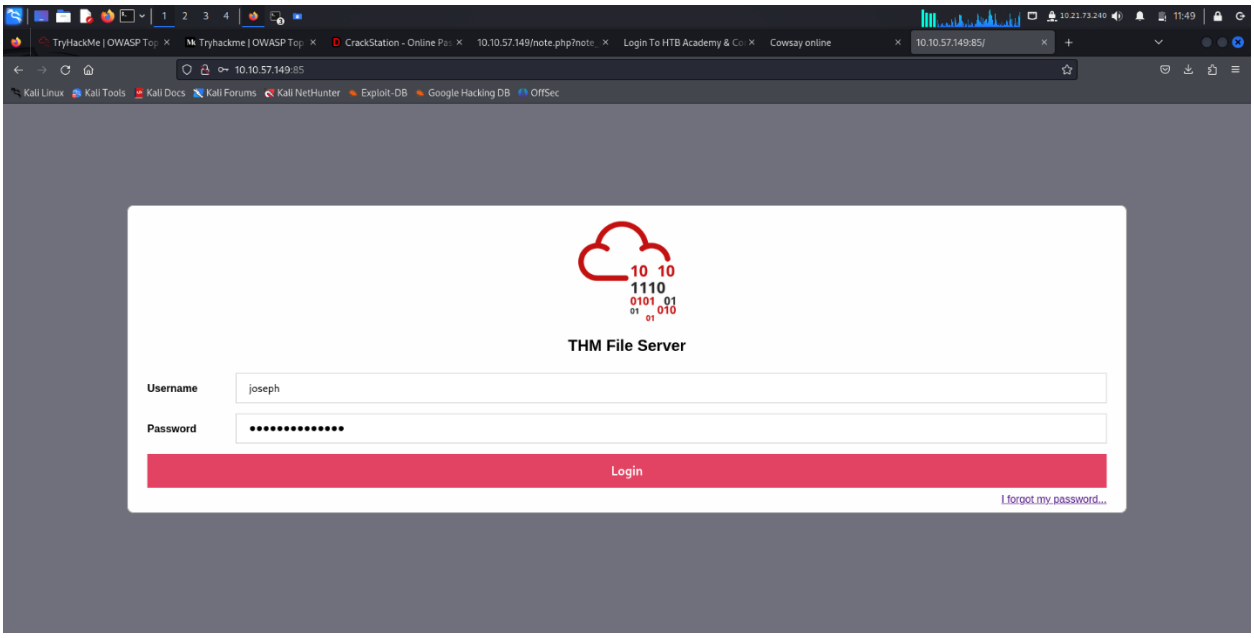


Without the password I couldn't login so I decided to try to rest the password, where I found security questions to answer, I picked the one that was easy to guess the answer, which was your favorite color? My lucky guess was green, which did the trick to reset the password automatically,

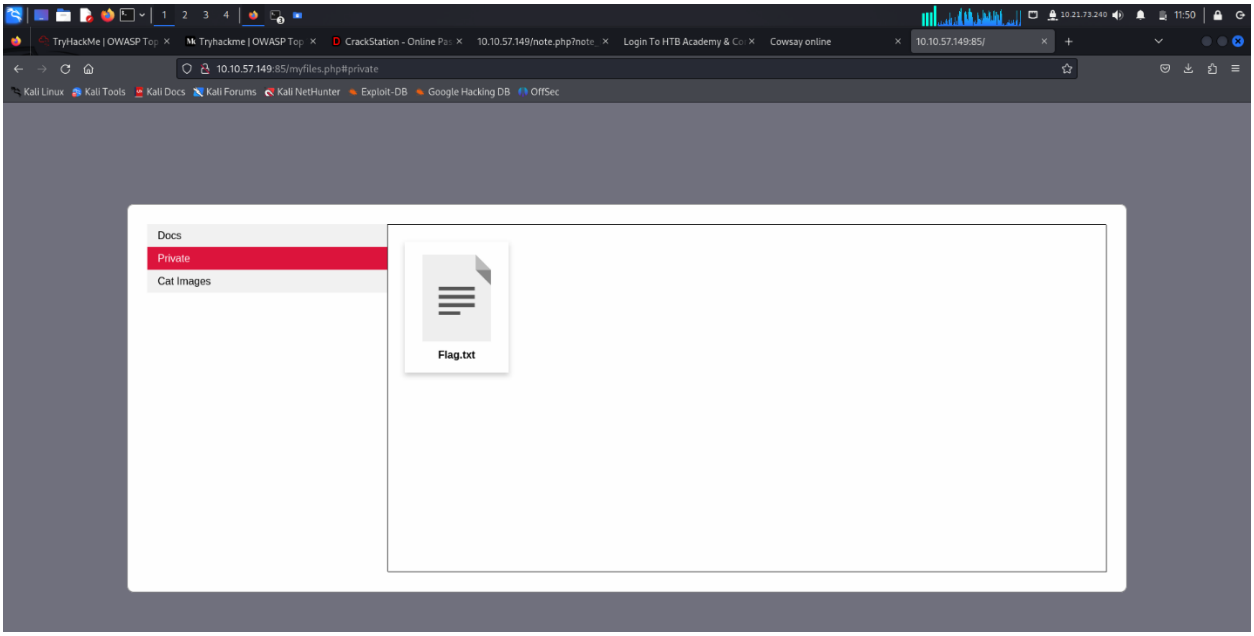

10.10.57.149:85/resetpass2.php
Password Reset
Step 2 - Please answer one of your security questions to confirm your identity:
Security Question
What's your favourite colour?
Answer
green
Continue


10.10.57.149:85/resetpass3.php
Password Reset
Success: The password for user joseph has been reset to DKuqtxed43Awvp
<< Back to Login

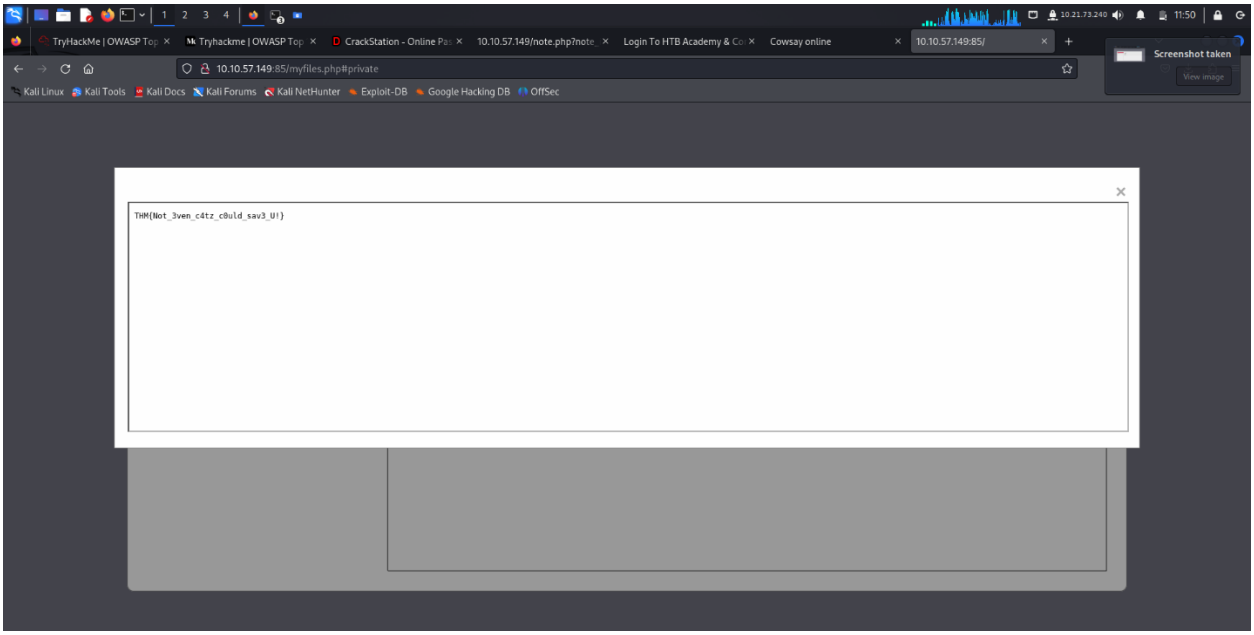
With the password generated and the username I had initially I tried to login again and it was a success,



Found in the user's page there was documents and images, in the private folder is where I found the flag file,



I opened the flag file and found the flag,



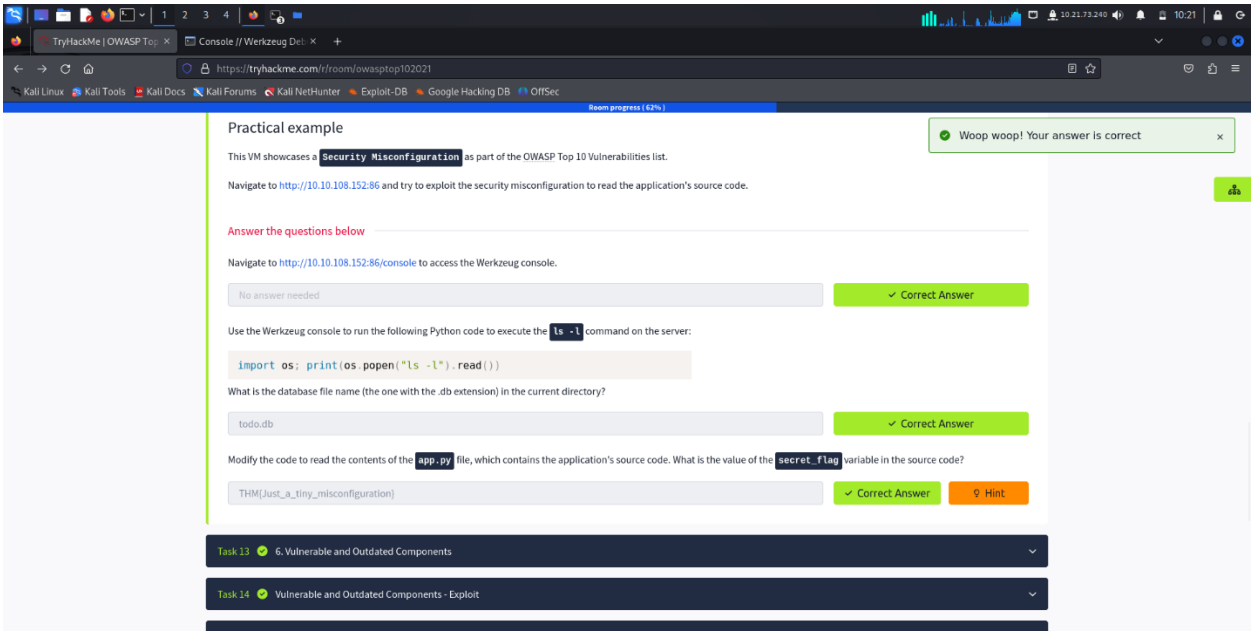
1.5Security Misconfigurations

Security Misconfigurations occur when security could have been appropriately configured but was not. Even if you download the latest up-to-date software, poor configurations could make your installation vulnerable.

Security Misconfigurations include, poorly configured permissions on cloud services, like S3 buckets, having unnecessary features enabled, like services, pages, accounts or privileges, default accounts with unchanged passwords, error messages that are overly detailed and allow attackers to find out more about the system and not using HTTP Security Headers.

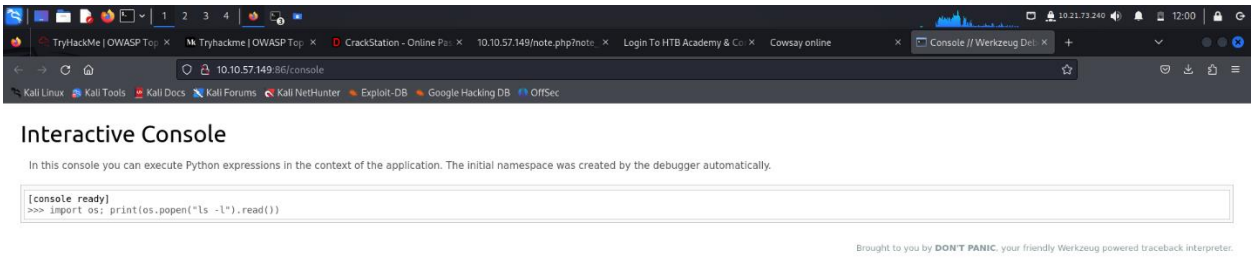
Debugging features is a common security misconfiguration and are often available in programming frameworks to allow the developers to access advanced functionality that is useful for debugging an application while it's being developed. Attackers could abuse some of those debug functionalities if somehow, the developers forgot to disable them before publishing their applications.

Questions

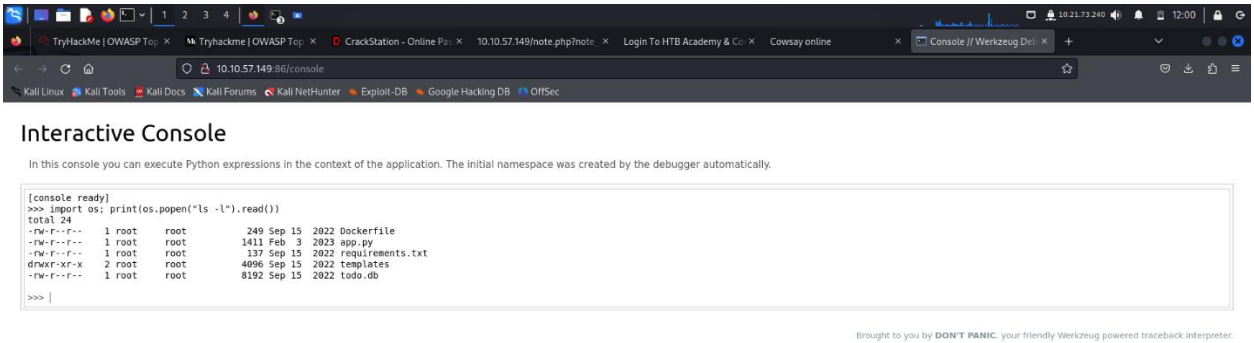


Quiz 1

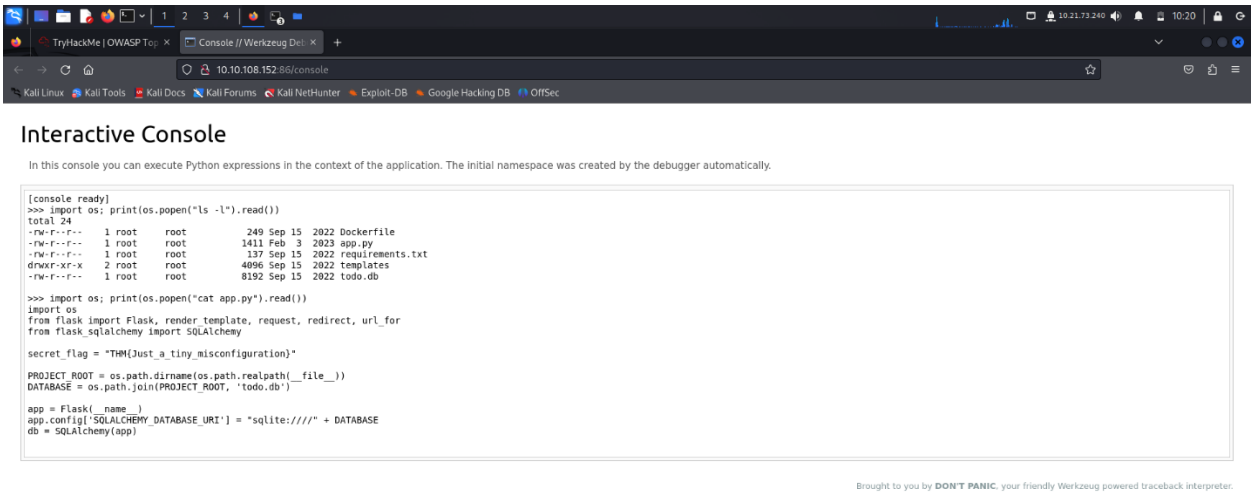
I loaded the target IP/console, and found console web platform where I could run python code, I used a python code and incorporated ls -l command to view the files,



This was the results amng them was a todo.db file, which was the answer



I modified the python code I used, by adding cat and file name, that I wanted to view its content, and found the flag,

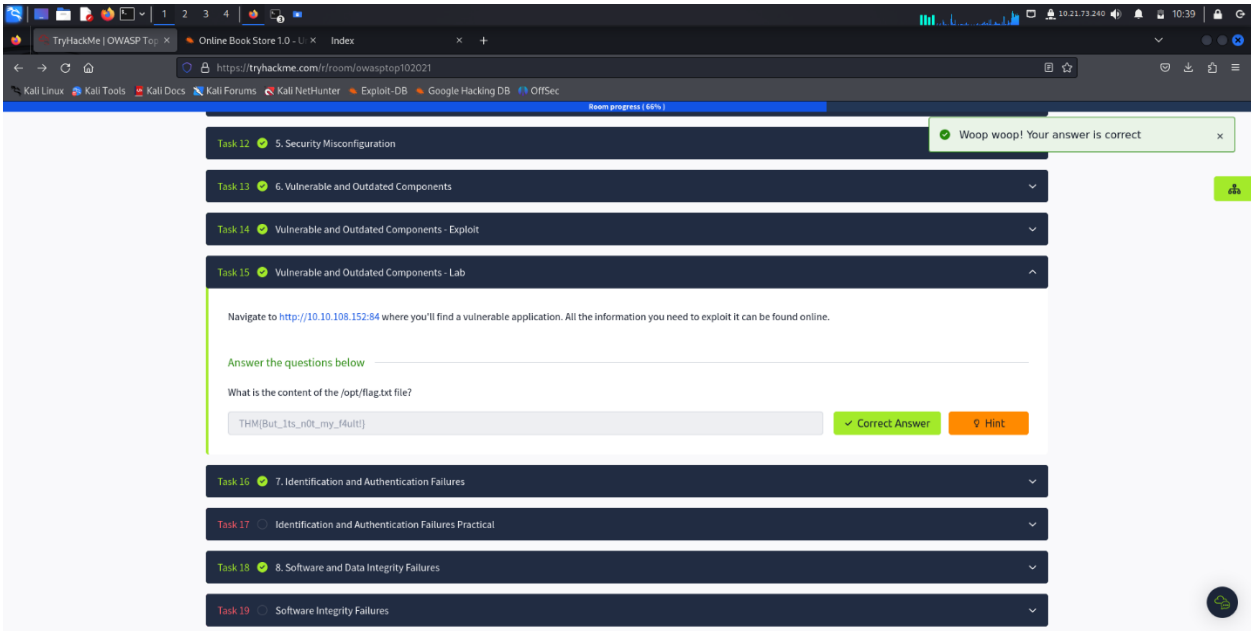


1.6Vulnerable and outdated Components

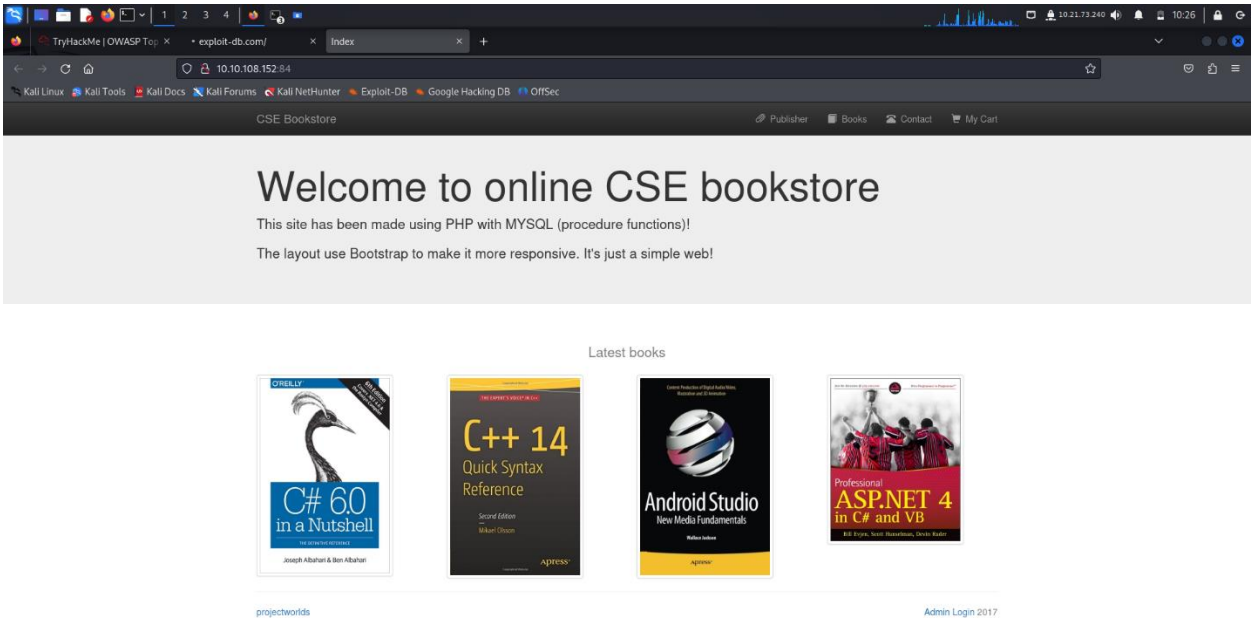
This vulnerability occurs when the systems are out-of-date making them to be vulnerable to exploits without the recent security patches to protect them.

Bet way to prevent this is to remove unused dependencies, unnecessary features, components, files and documentation. Doing continuous inventory of both the client-side and server-side components.

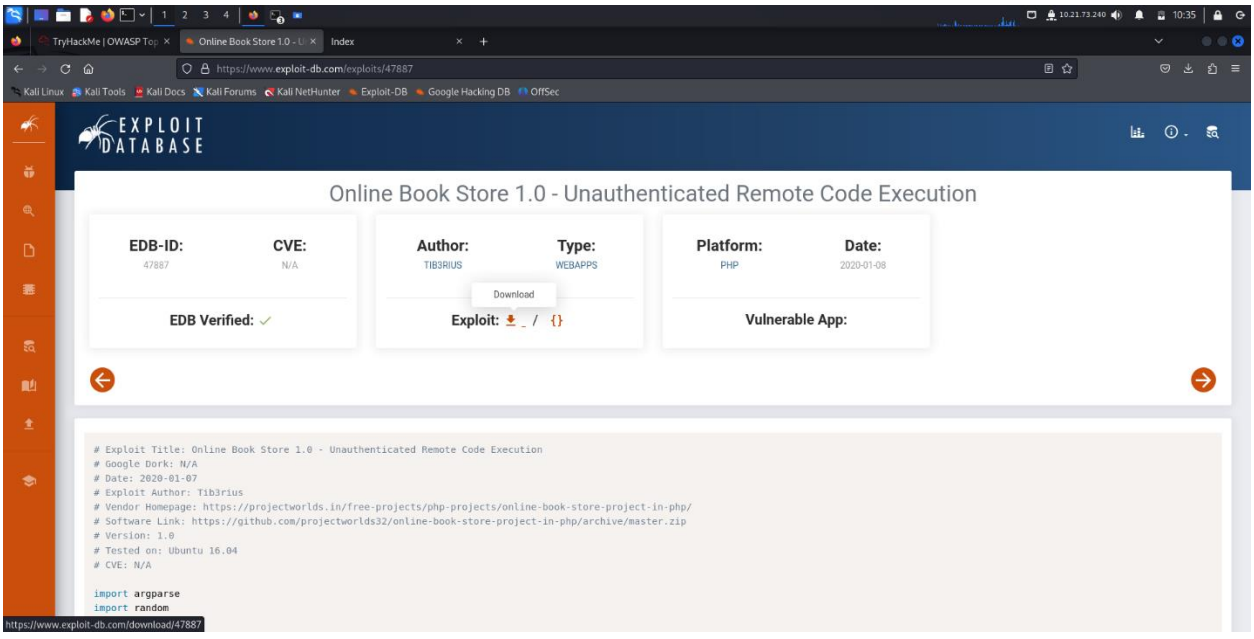
Questions



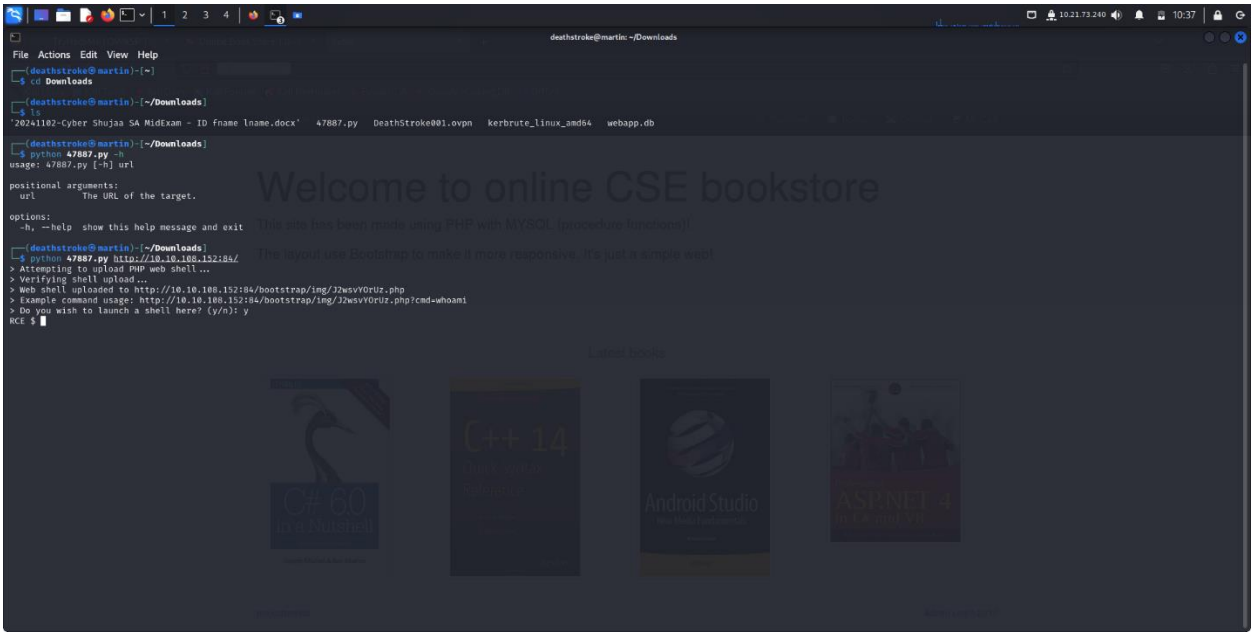
I launched the target IP and found it was a online bookstore platform,



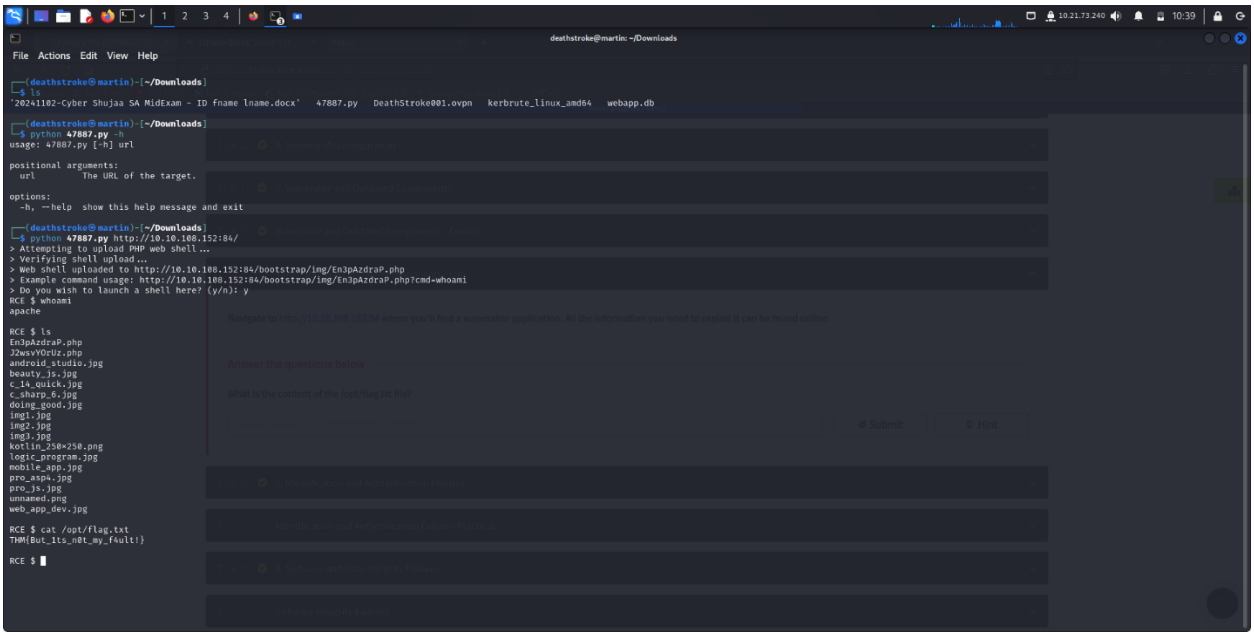
I used exploit DB platform to find exploit modules for the online bookstore platform, and I found unauthenticated remote code execution, I downloaded exploit module file, which was in the format of python,



After downloading, launched the exploit using python tool the exploit module, I downloaded, and the URL of the online bookstore, and I was able to get a shell back,



From the shell used command whoami to view the username of the machine to be apache, using cat and path /opt/flag.txt I was able to view the contents of the flag file, and found the flag,



1.7Identifications and authentication failures

Authentication allows users to gain access to web applications by verifying their identities. The most common form of authentication is using a username and password mechanism and server would then provide the users' browser with a session cookie (if they are correct), needed because web servers use HTTP(S) to communicate, which is stateless. Attaching session cookies means the server will know who is sending what data. The server can then keep track of users' actions.

Some common flaws in authentication mechanisms include the following;
Brute force attacks, if a web application uses usernames and passwords, an attacker can try to launch brute force attacks that allow them to guess the username and passwords using multiple authentication attempts.

Use of weak credentials, Web applications should set strong password policies. If applications allow users to set passwords such as "password1" or common passwords, an attacker can easily guess them and access user accounts.

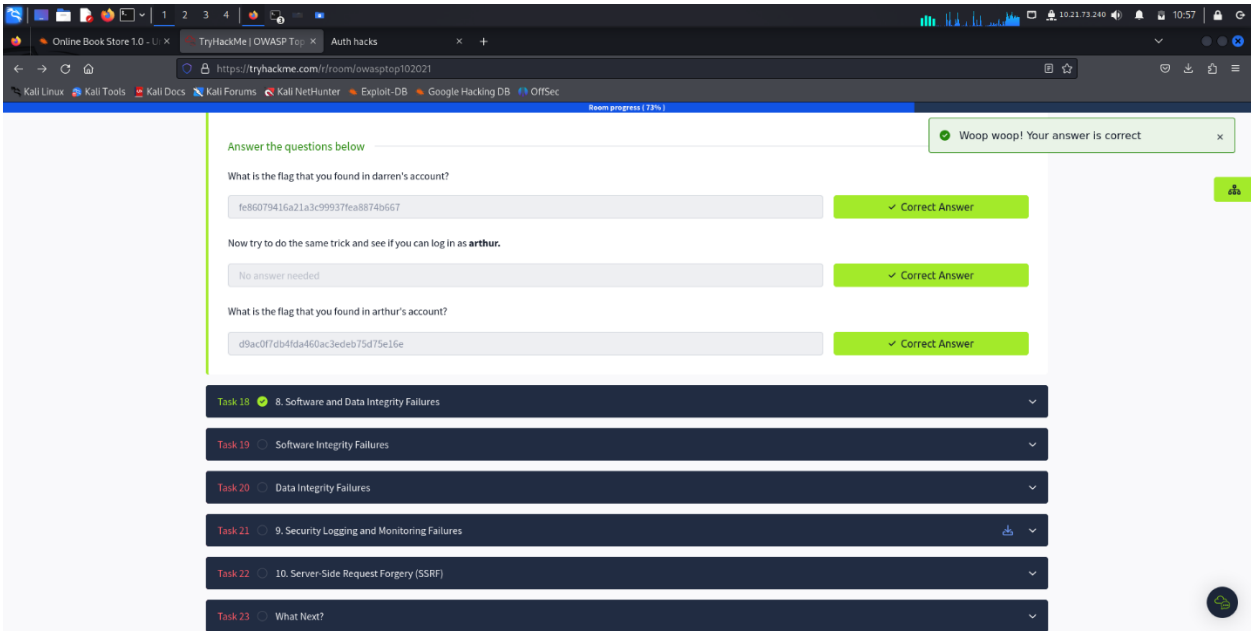
Weak Session Cookies are how the server keeps track of users. If session cookies contain predictable values, attackers can set their own session cookies and access users' accounts.

Best way to prevent this is through avoid password-guessing attacks, ensure the application enforces a strong password policy.

Avoid brute force attacks, ensure that the application enforces an automatic lockout after a certain number of attempts. This would prevent an attacker from launching more brute-force attacks.

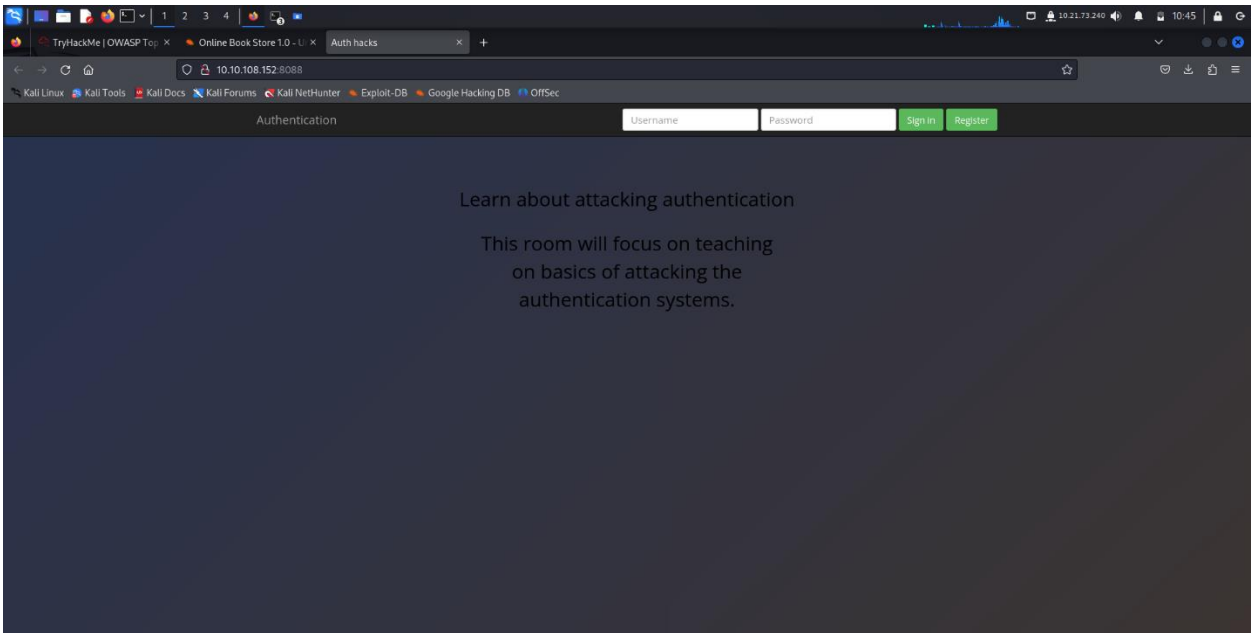
Implementing Multi-Factor Authentication.

Questions

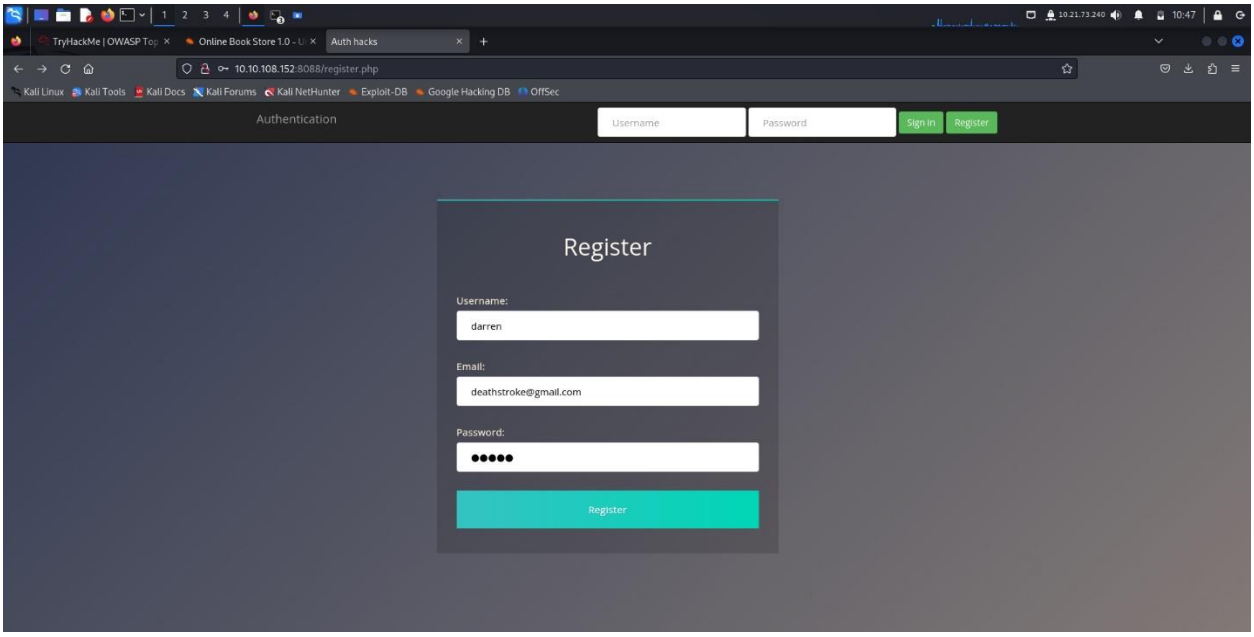


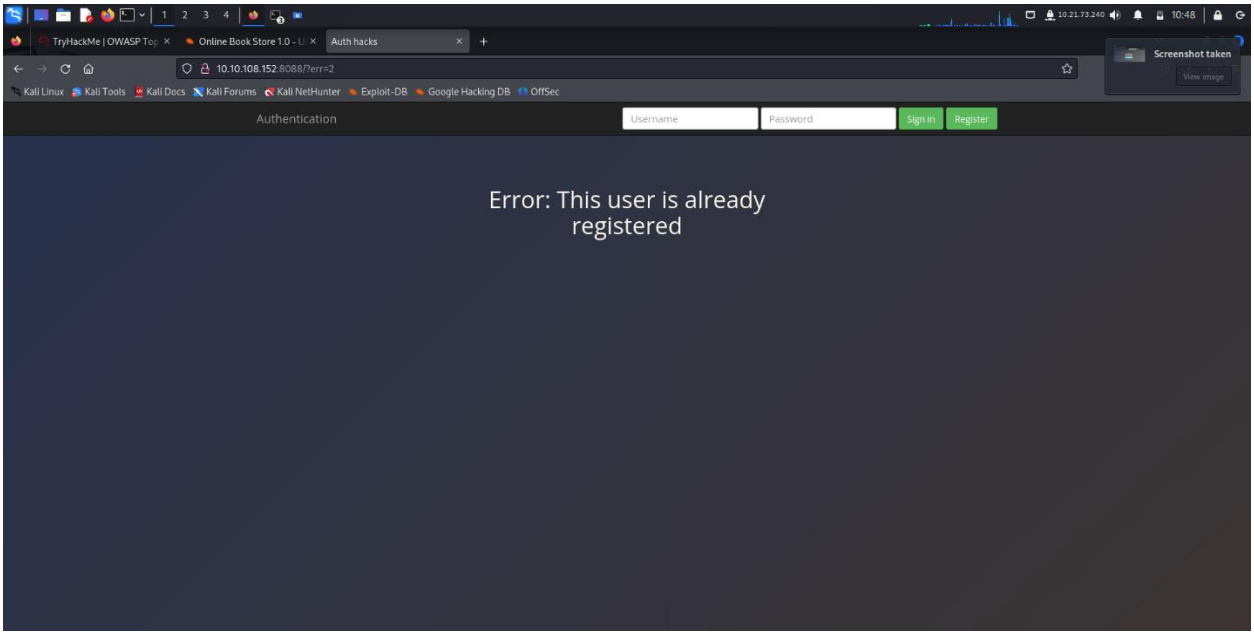
Quiz 1

I launched the site using the target IP, and this how the platform looked like,



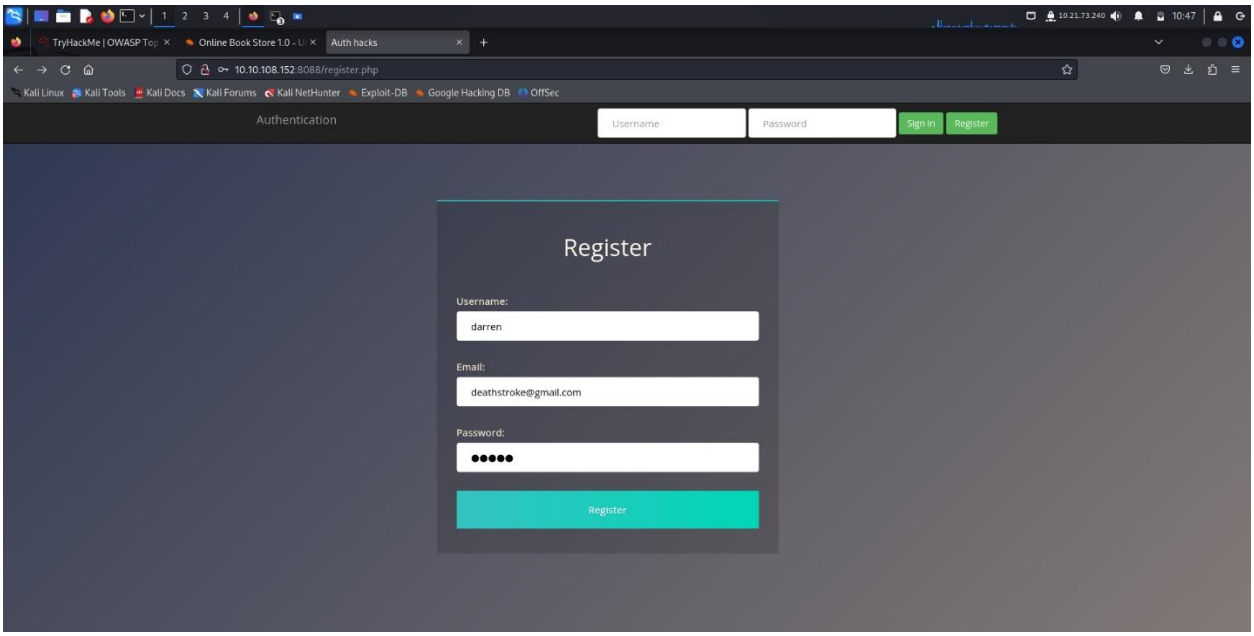
Used the register button to try to register a user, tried to see if a user Darren was already registered, and this was a the results, I found he was already registered,

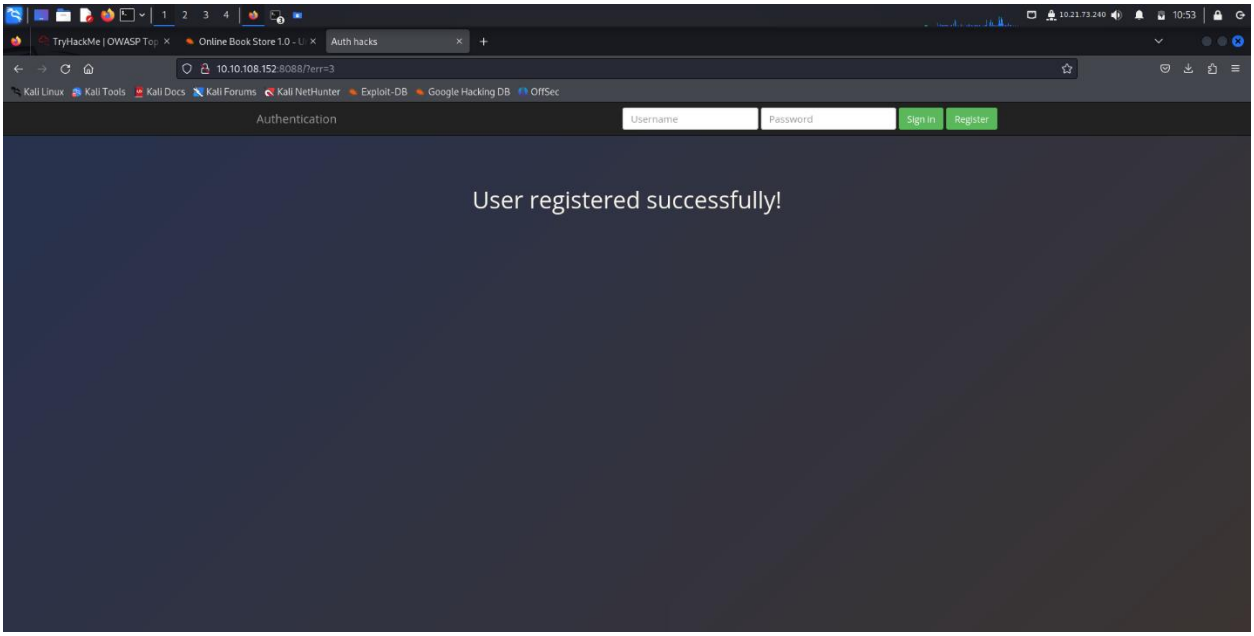




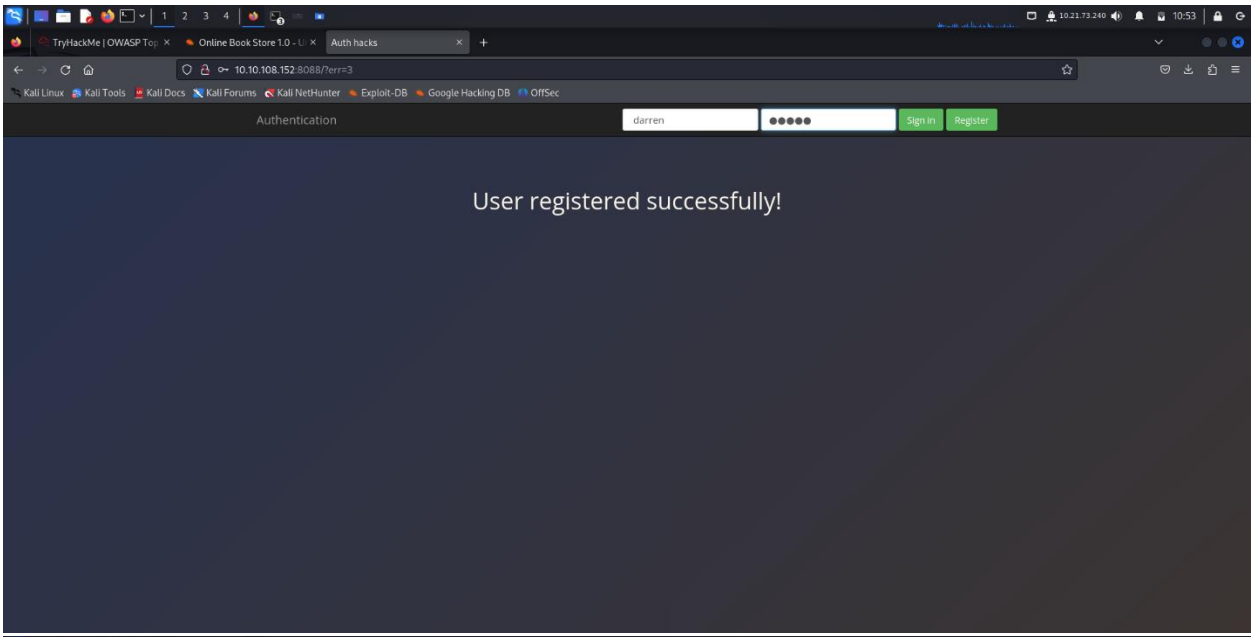
After finding out the user was already registered, I tried to check whether I could register a user with same username and different password, so as to see if this new user will have same rights as Darren,

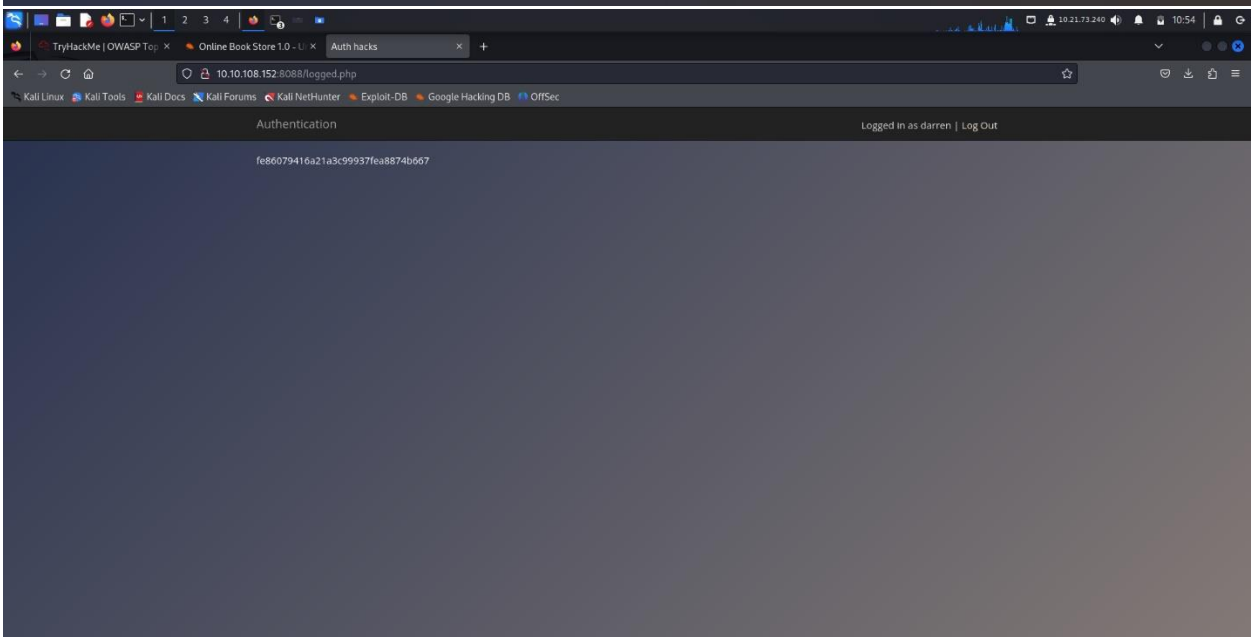
On the username section I added a space then followed by the name, I the registration was successful,





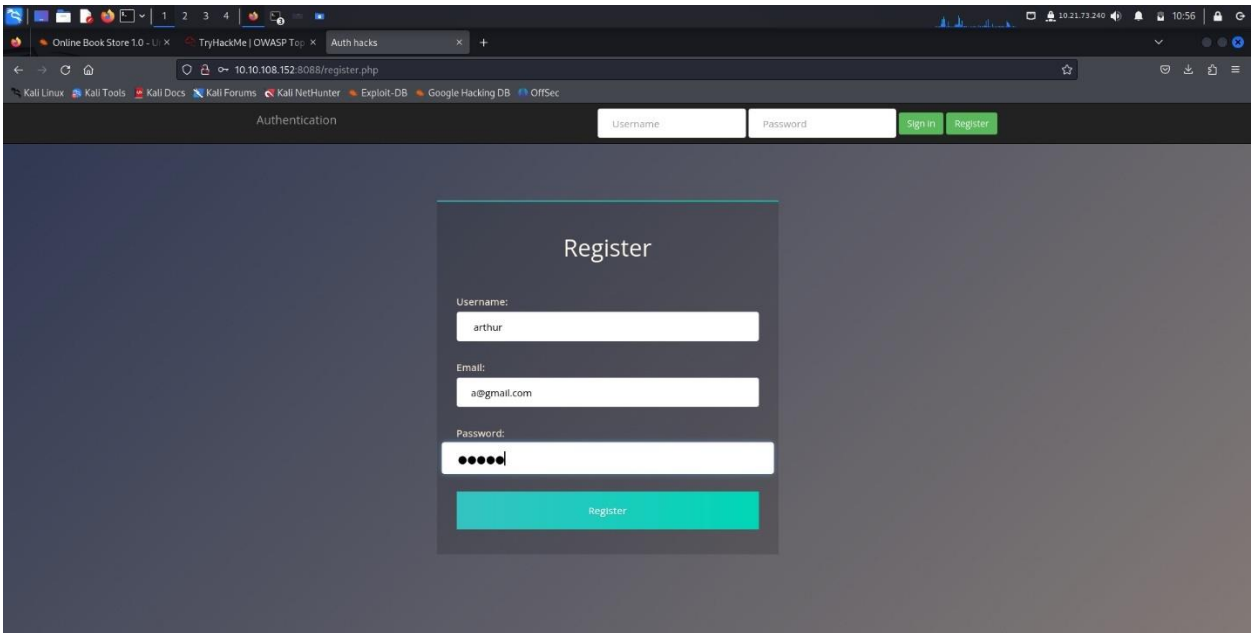
From the new user I created I tried to login, with the username and the password and it was successful,

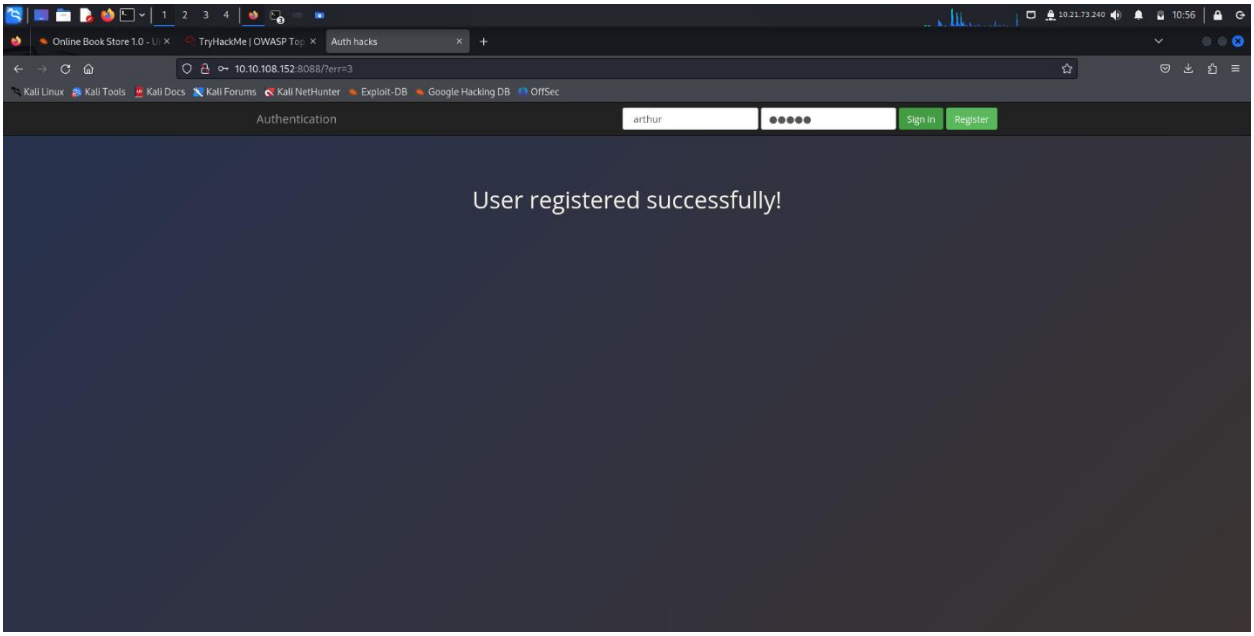




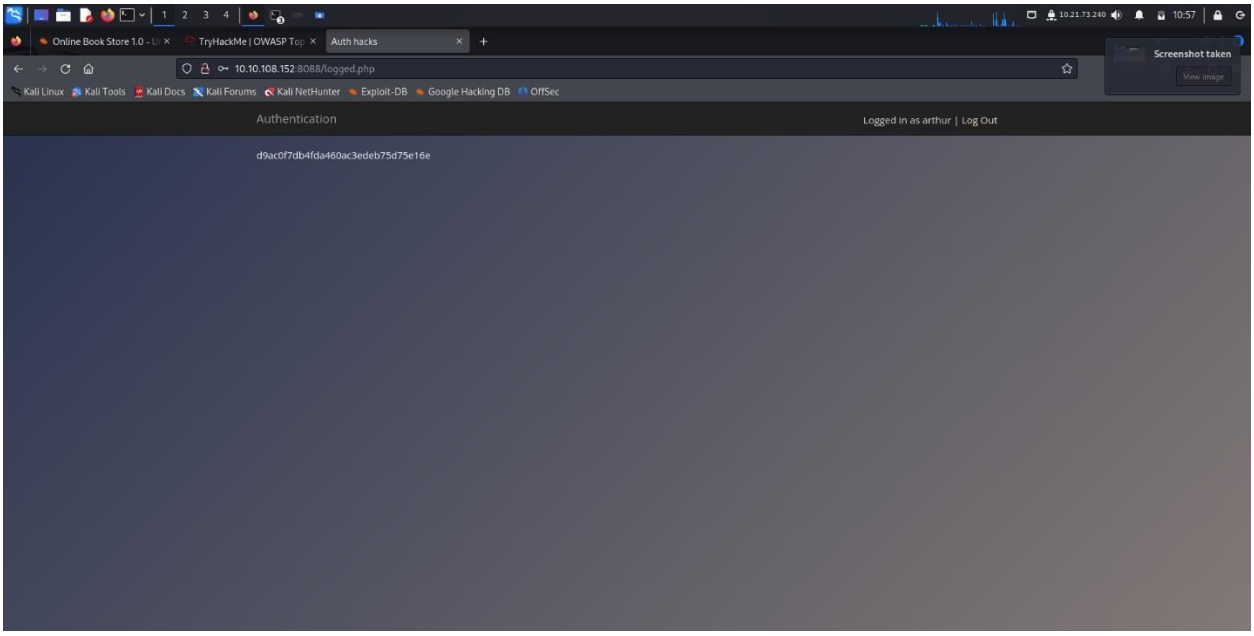
Quiz 2

I tried to do the same to user Arthur, by creating another user with username Arthur, in this case the username started with a space and then username(Arthur), and created my own password and it was successful,





I logged in with username arthur and a password and found the flag,ss



1.8Software and Data Integrity failures

This vulnerability arises from code or infrastructure that uses software or data without using any kind of integrity checks. Since no integrity verification is being done, an attacker might modify the software or data passed to the application, resulting in unexpected consequences. And it can either be software Integrity Failures or Data Integrity Failures.

To prevent this type of attack is best to use digital signatures, or even cryptographic algorithms to encrypt the data.

Software Integrity Failures

Scenario

Suppose you have a website that uses third-party libraries that are stored in some external servers that are out of your control. While this may sound a bit strange, this is actually a somewhat common practice. With an example of jQuery, a commonly used javascript library. If you want, you can include jQuery in your website directly from their servers without actually downloading it by including the following line in the HTML code of your website: <script src="https://code.jquery.com/jquery-3.6.1.min.js"></script>

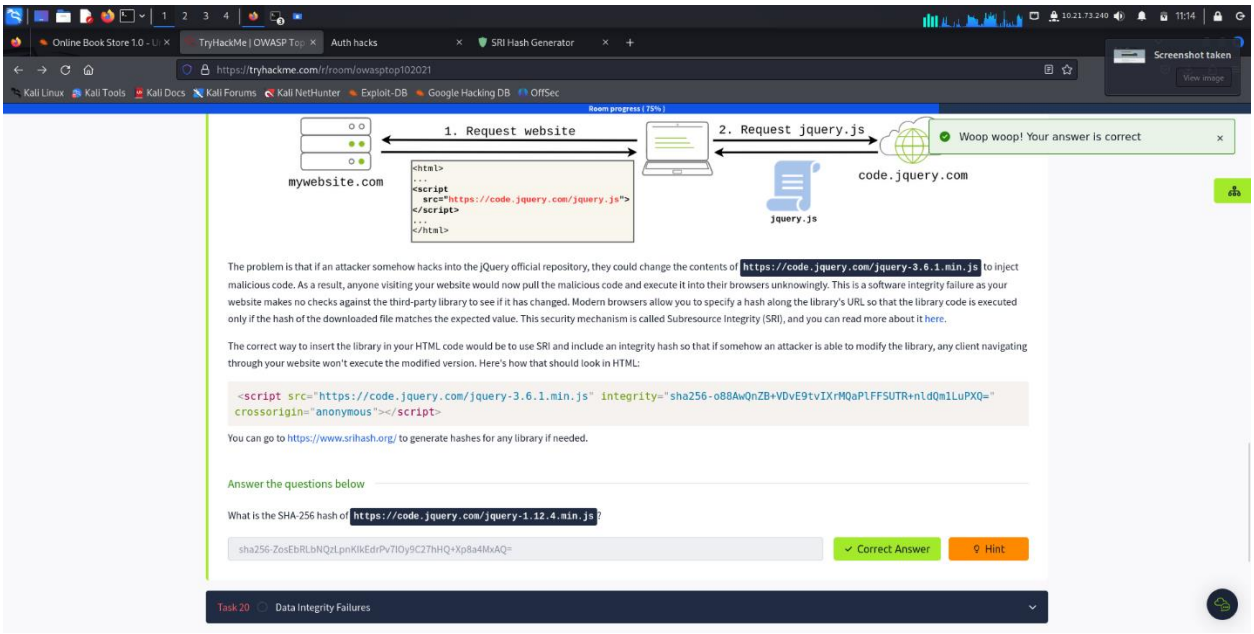
When a user navigates to your website, its browser will read its HTML code and download jQuery from the specified external source.

The problem is that if an attacker somehow hacks into the jQuery official repository, they could change the contents of https://code.jquery.com/jquery-3.6.1.min.js to inject malicious code. As a result, anyone visiting your website would now pull the malicious code and execute it into their browsers unknowingly. This is a software integrity failure as your website makes no checks against the third-party library to see if it has changed.

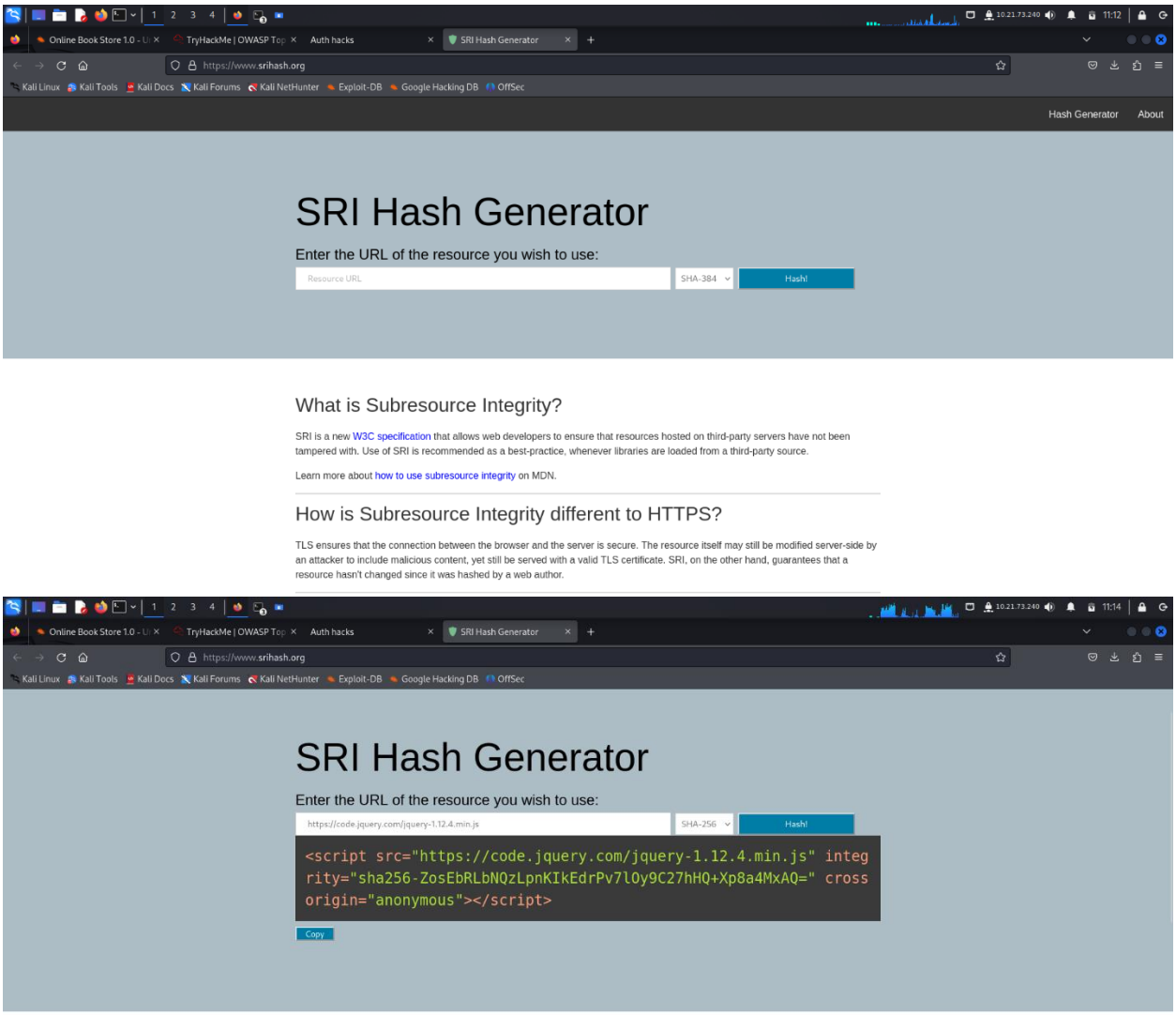
Modern browsers allow you to specify a hash along the library's URL so that the library code is executed only if the hash of the downloaded file matches the expected value. This security mechanism is called Subresource Integrity (SRI).

<script src="https://code.jquery.com/jquery-3.6.1.min.js" integrity="sha256-o88AwQnZB+VDvE9tvIXrMQaPlFFSUTR+nldQm1LuPXQ=" crossorigin="anonymous"></script> this is correct way to insert a library into HTML Code

Questions



I used platform SRI Hash generator to generate an encryption of this library(https://code.jquery.com/jquery-1.12.4.min.js) that was to be inserted to a HTML code,



Data integrity failures

when a user logs into an application, they will be assigned some sort of session token that will need to be saved on the browser for as long as the session lasts. This token will be repeated on each subsequent request so that the web application knows who we are. These session tokens can come in many forms but are usually assigned via cookies.

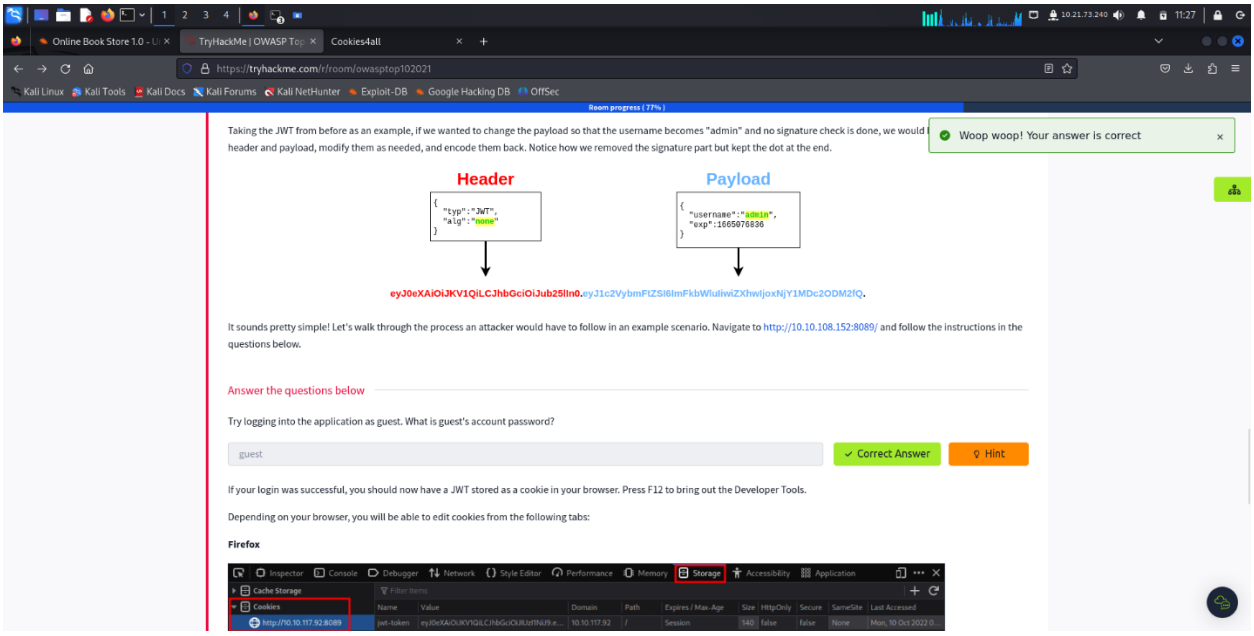
Scenario

webmail application, you could assign a cookie to each user after logging in that contains their username. This would be a terrible idea security-wise because cookies are stored on the user's browser, so if the user tampers with the cookie and changes the username, they could potentially impersonate someone else and read their emails! This application would suffer from a data integrity failure.

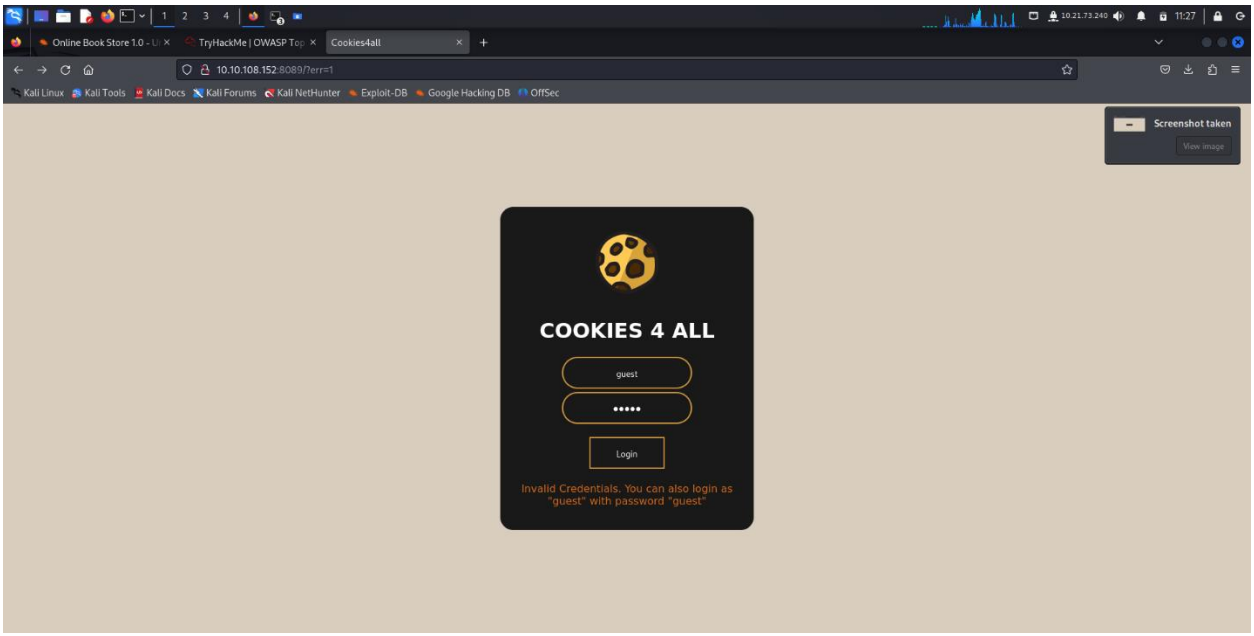
One solution to this is to use some integrity mechanism to guarantee that the cookie hasn't been altered by the user, with implementation of JSON Web Tokens (JWT).

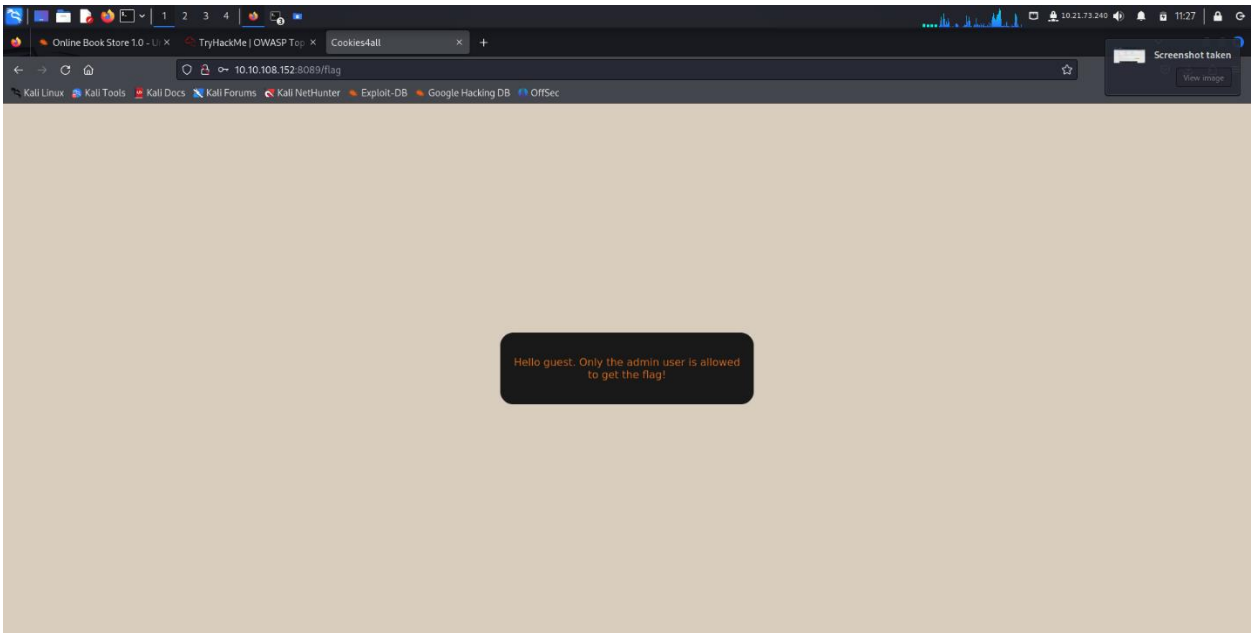
JWTs are very simple tokens that allow you to store key-value pairs on a token that provides integrity as part of the token. The idea is that you can generate tokens that you can give your users with the certainty that they won't be able to alter the key-value pairs and pass the integrity check. Has three parts Header, payload, signature,

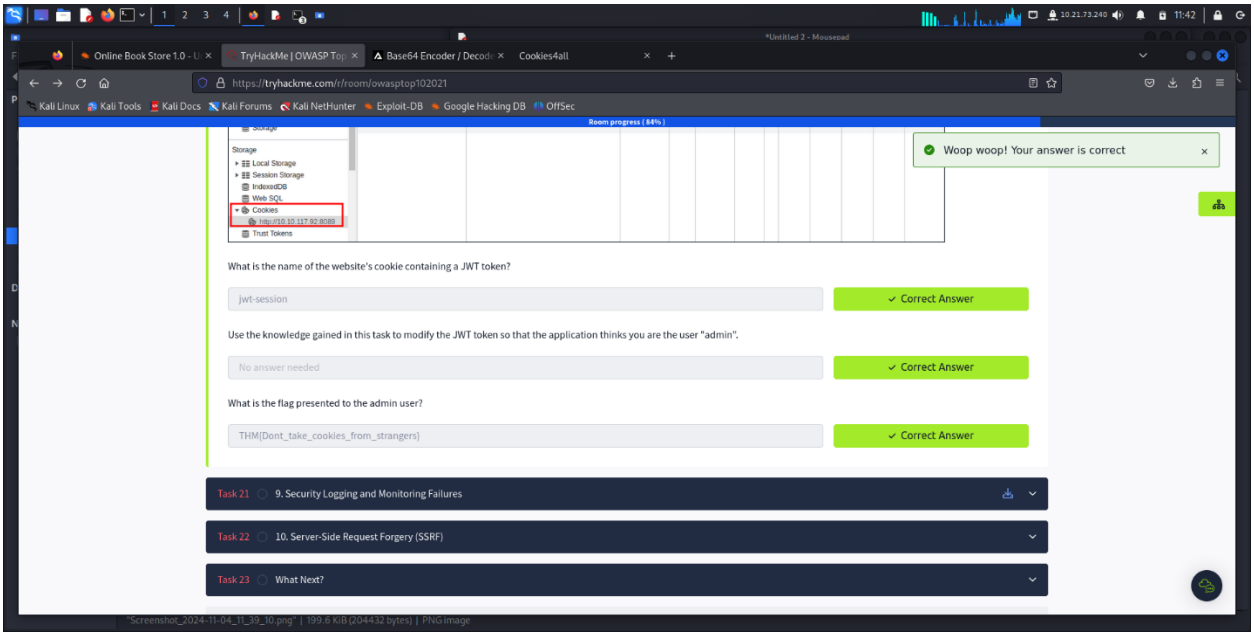
Questions



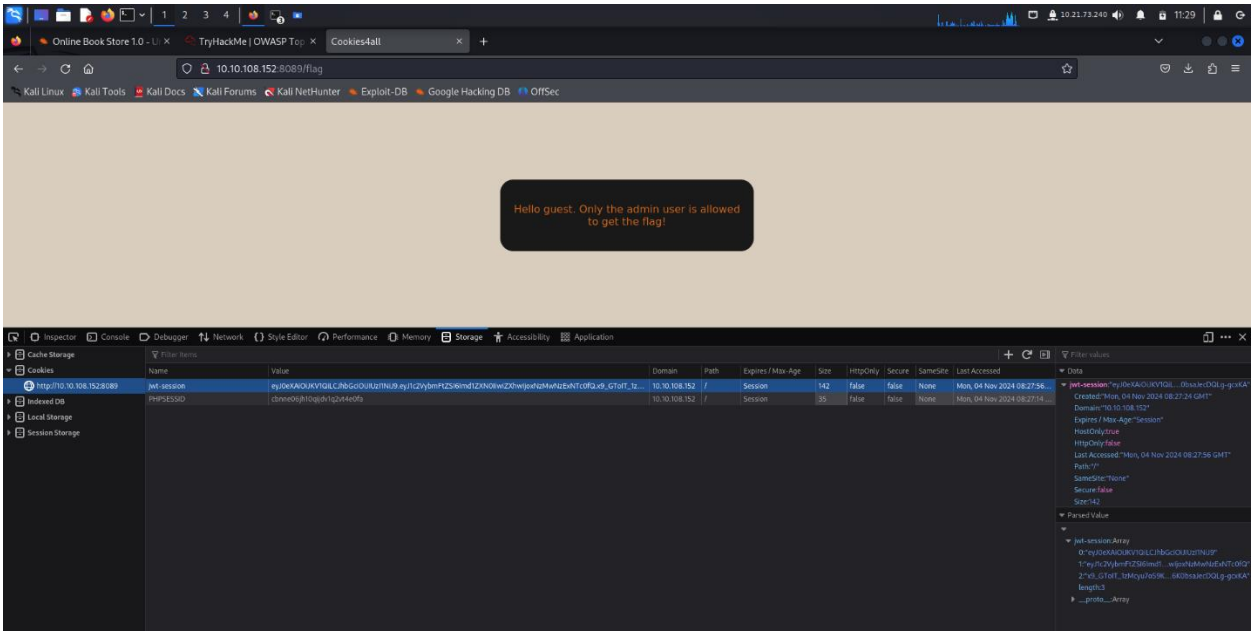
I launched the site using the target IP and found it was a login page, I had been given a username(guest) and was to find the password, I tried guessing the password to guest and it was that, used it and successfully login.







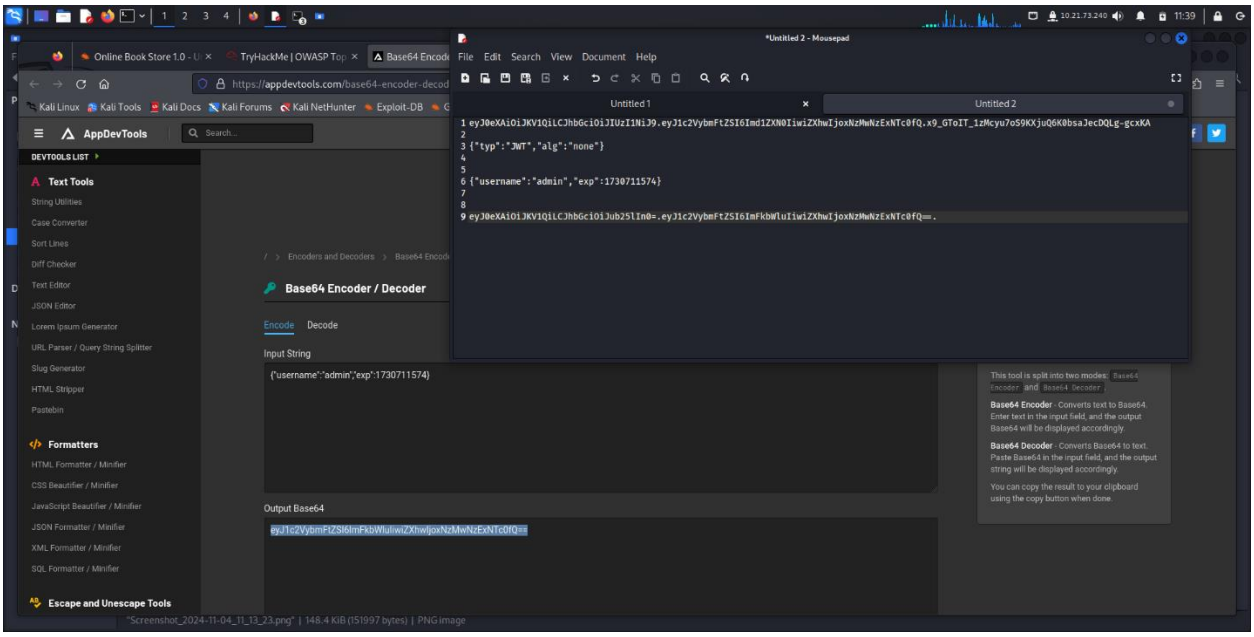
After successful login, inspected the page, and found the name of the cookie that was being used for the session,



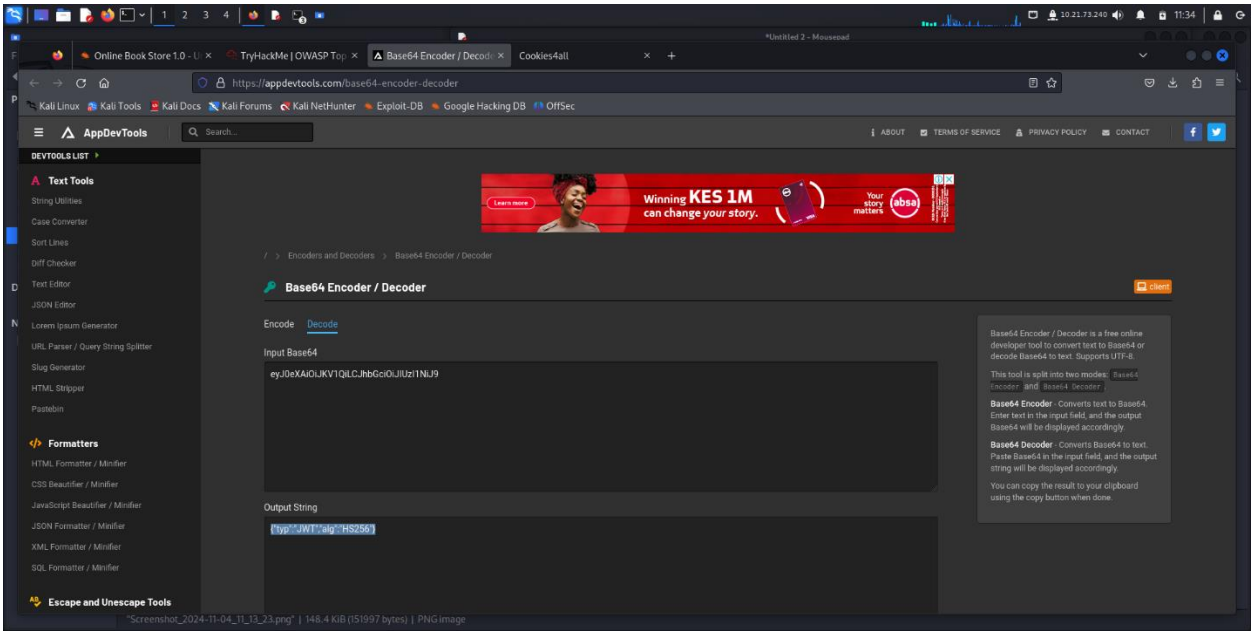
Quiz 2

After find the cookie it was time to start decoding the sections of the cookie, to try to modify the decode part and later encode it again to try to find a cookie to use to gain access to a session an admin,

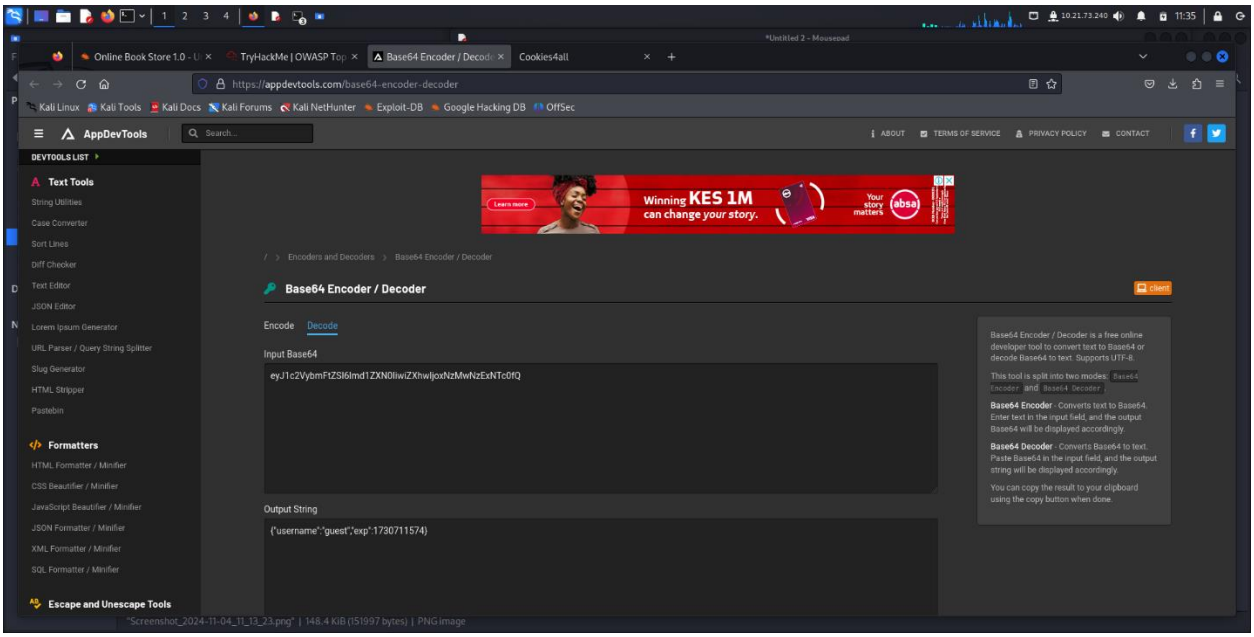
I started by first decoding the first part of cookie and modifying it, and decoding the second part of the cookie and modifying it,



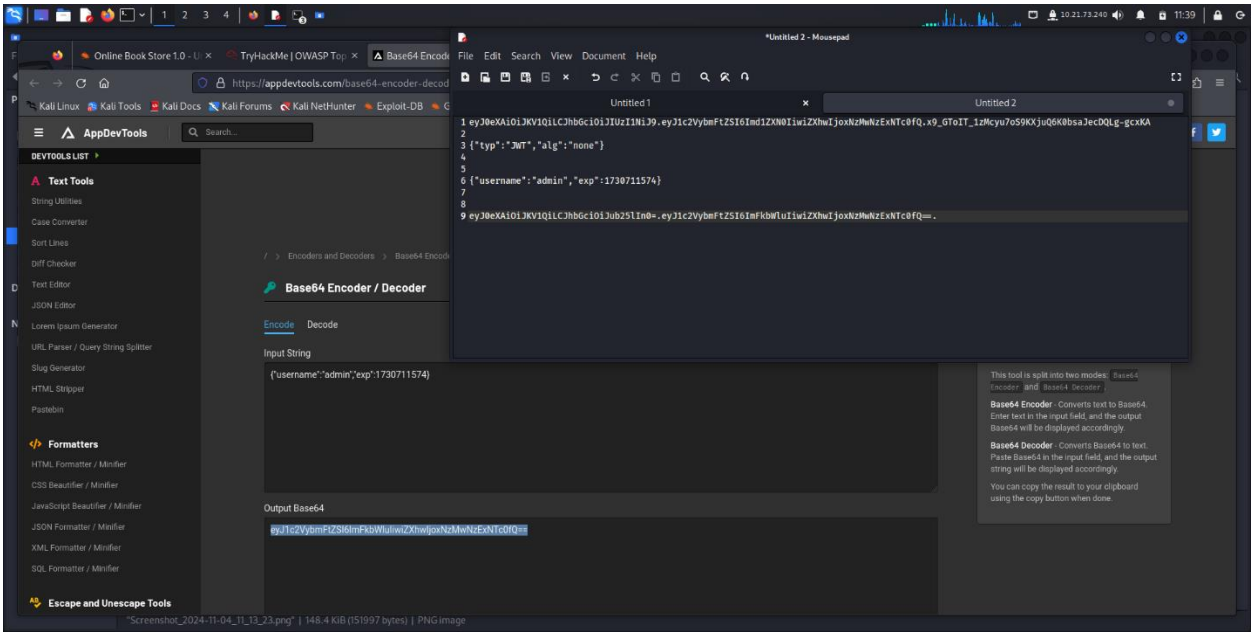
This how I decode the first part of the cookie using a appdevtools.com platform, after decoding I modified it to have no cryptographic encryption algorithm,



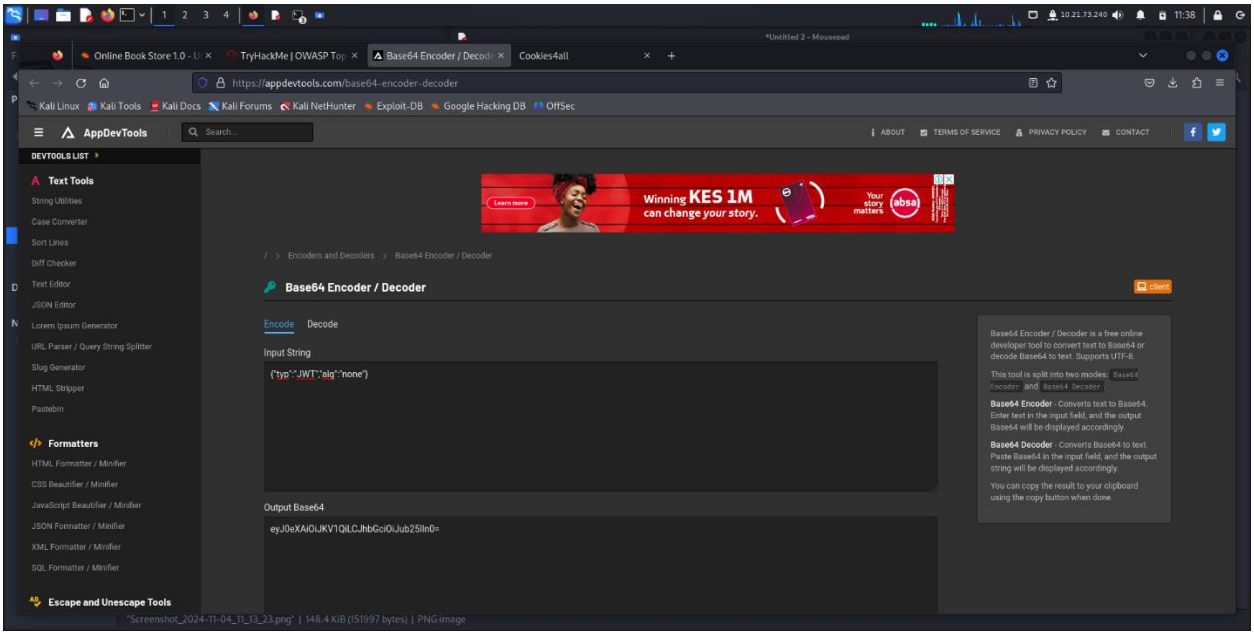
This how I decoded the second part of the cookie, which had he username guest, I modified it to have username admin,



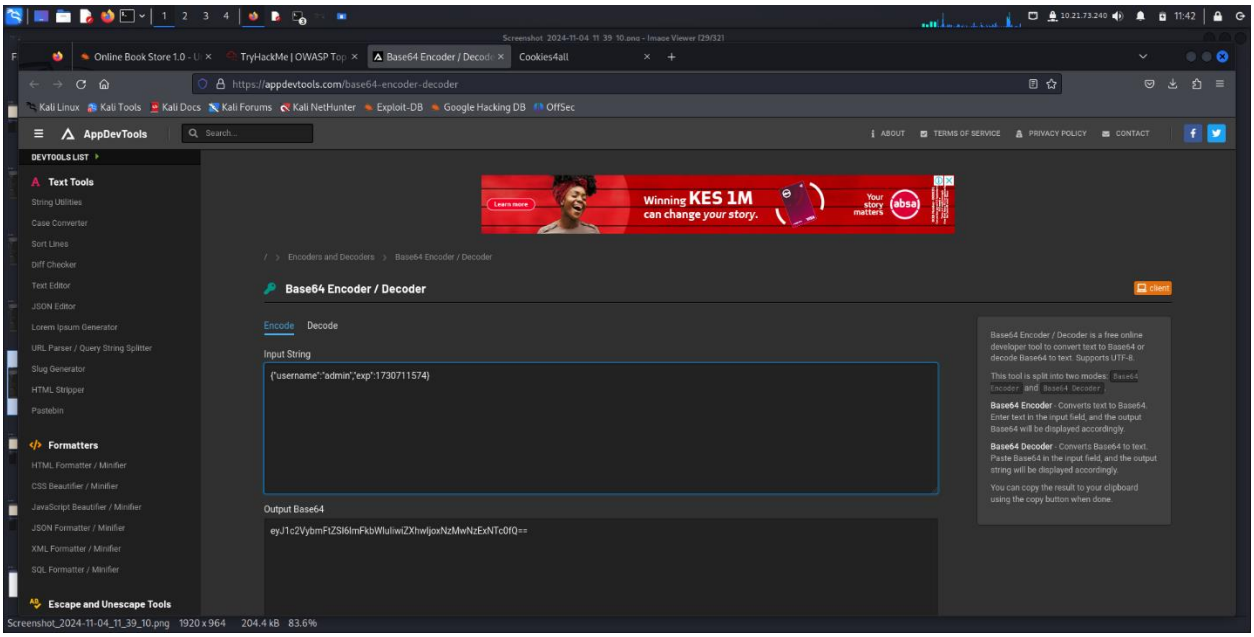
In this image you can see the modifications I was talking about, no hash algorithm and username(admin), after all this modification I had to encode it, to come up with final cookie,



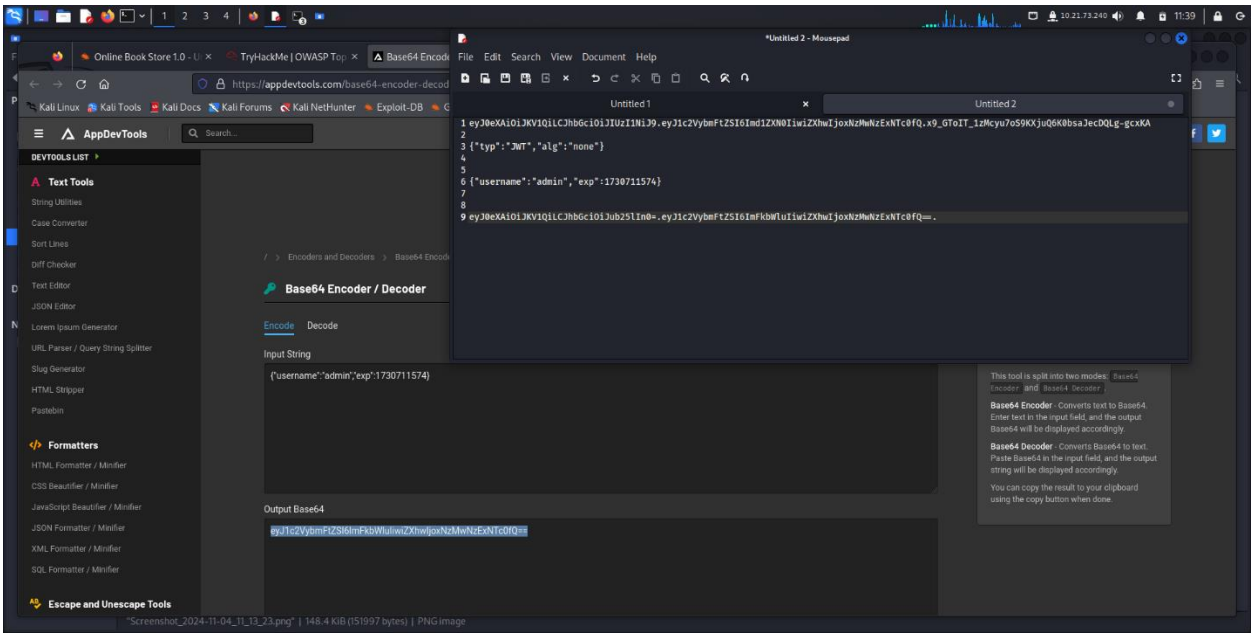
Stared by encoding the first part of cookie that I modified,



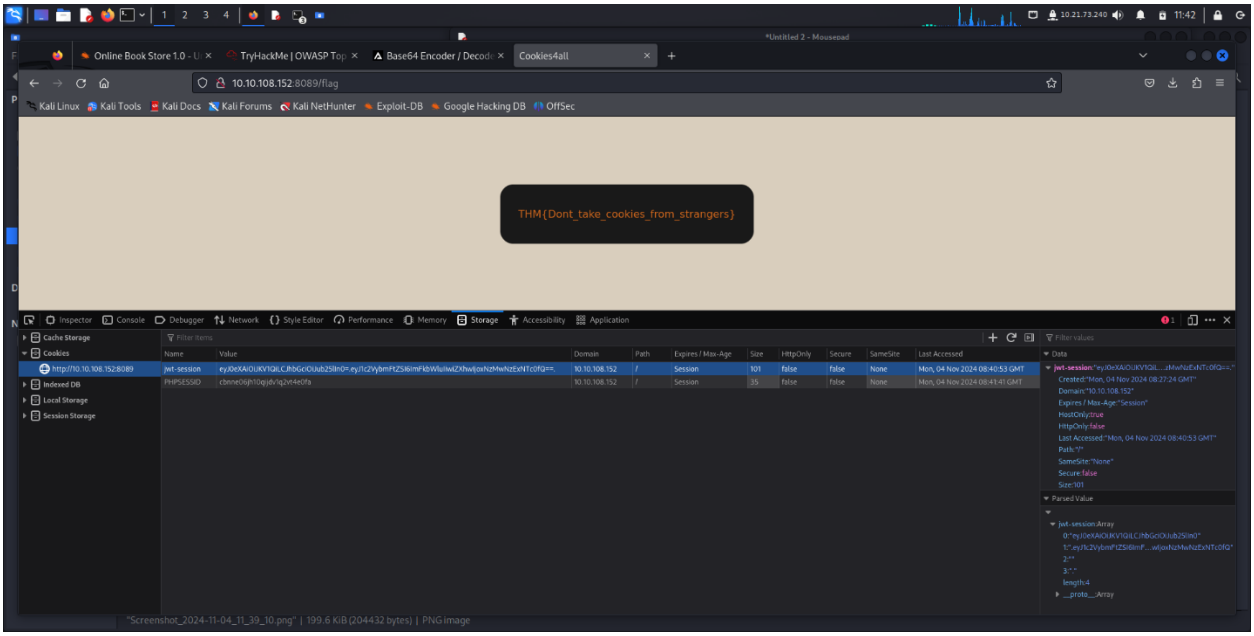
Encoded the second part of the cookie,



In this image, you can see the first encoded part, combined with the second encoded part, to for a cookie that I used to gain access to a session as admin,



With the new cookie created, I replaced the old cookie with this cookie, and reloaded the page where now I had gain access as an admin and found the flag,

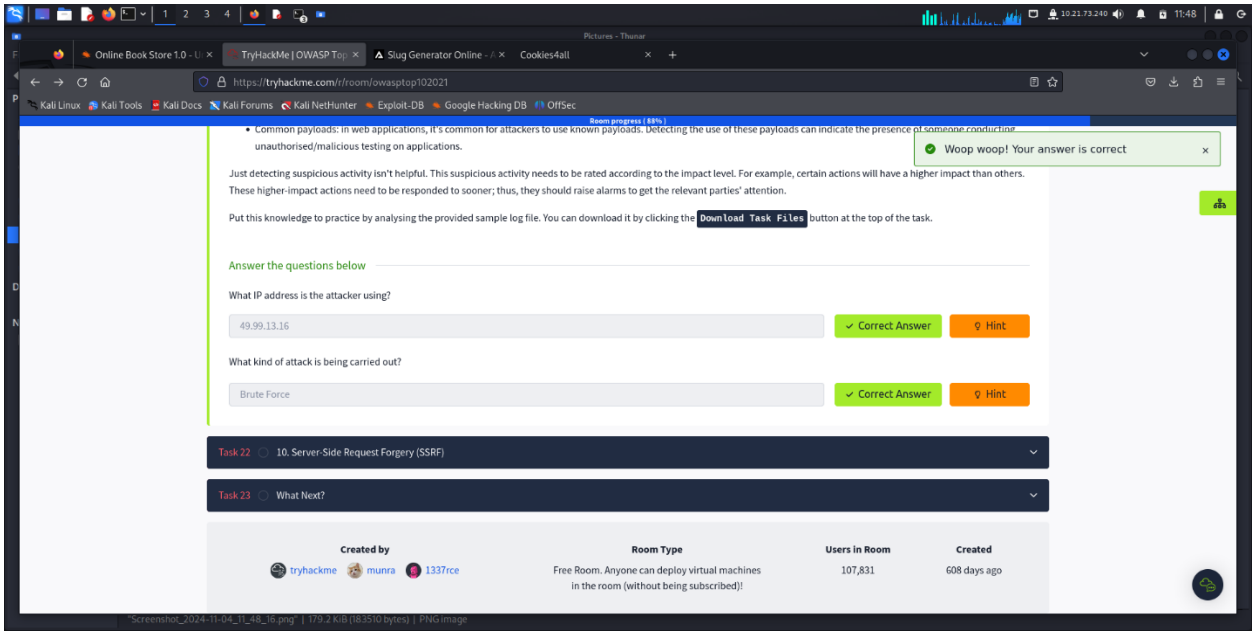


1.9Security Logging and Monitoring Failures

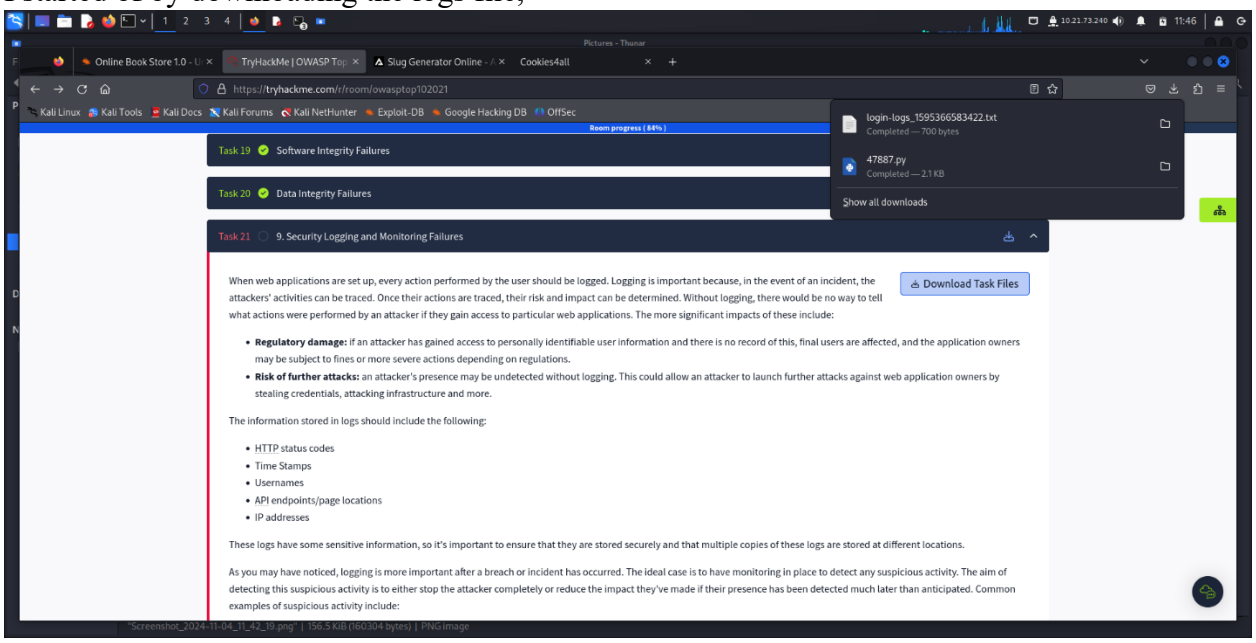
When web applications are set up, every action performed by the user should be logged. Logging is important because, in the event of an incident, the attackers' activities can be traced. Once their actions are traced, their risk and impact can be determined. Without logging, there would be no way to tell what actions were performed.

This results to Regulatory damage, Risk of further attacks.

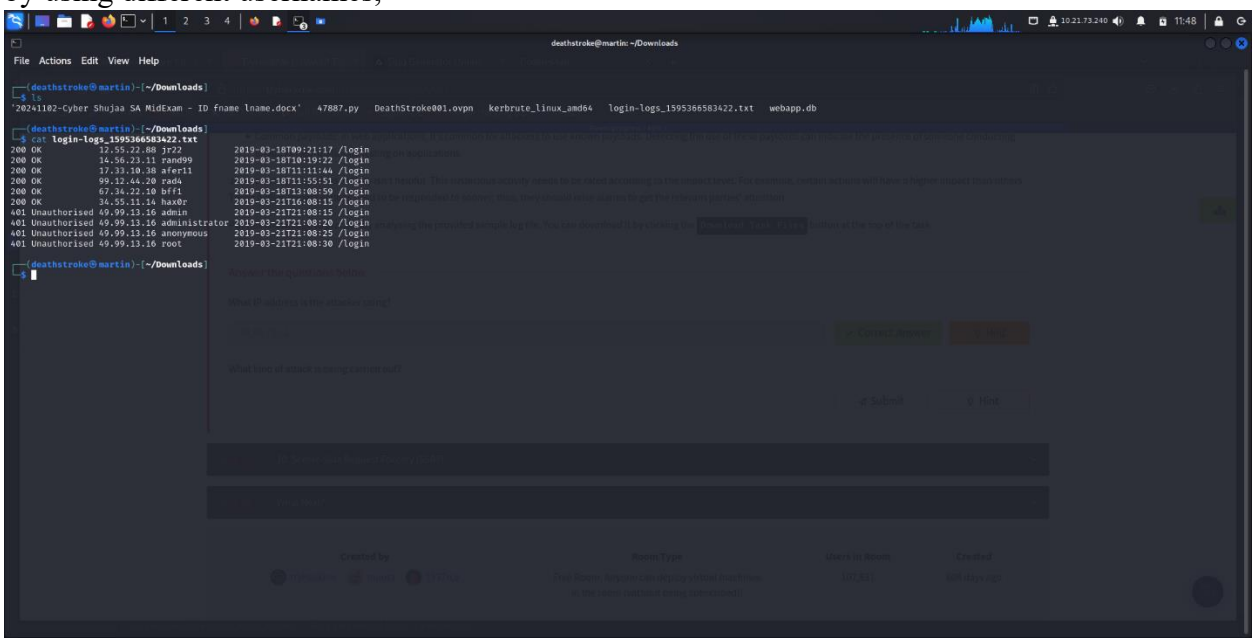
Questions



I started of by downloading the logs file,



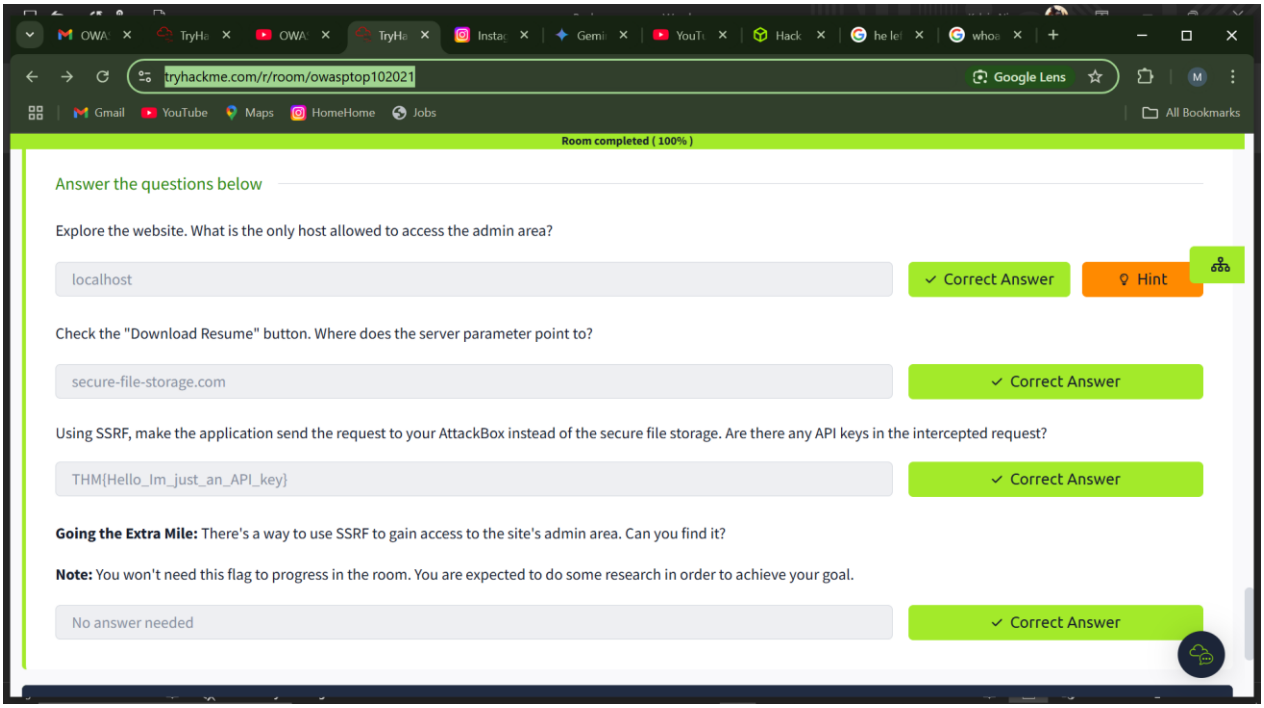
After successful download I used cat to view the contents of the logs file, and found there were users trying to login and among them was An IP that was trying to brute force its login, by using different usernames,



1.10 Server-Side Request Forgery (SSRF)

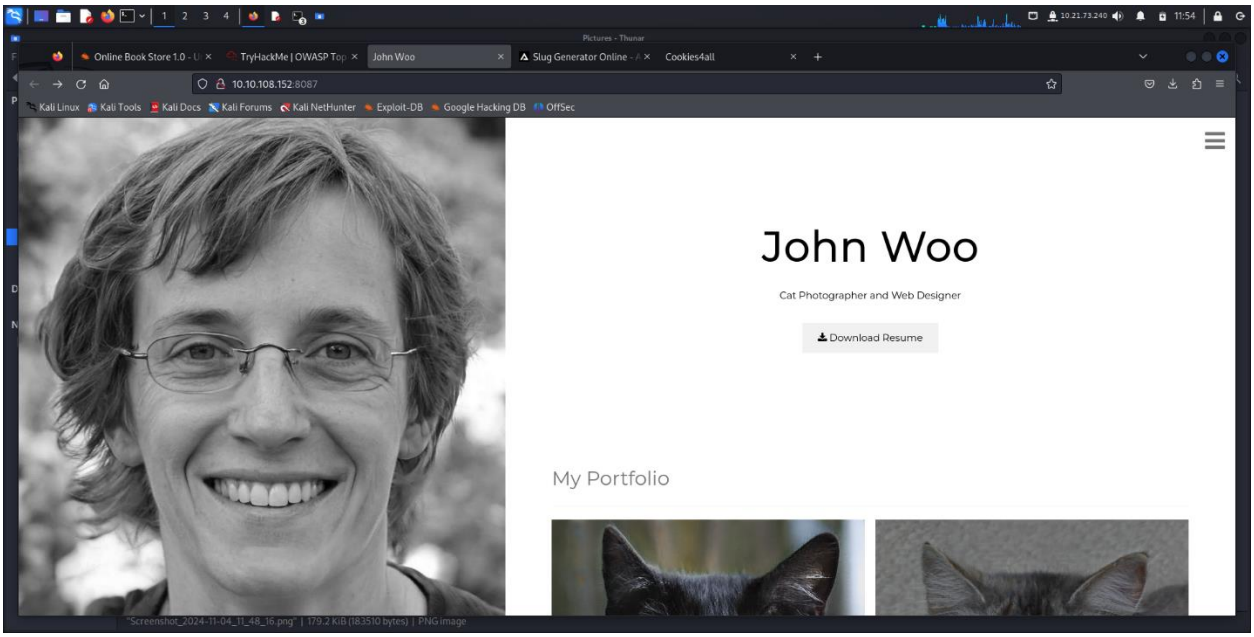
This type of vulnerability occurs when an attacker can coerce a web application into sending requests on their behalf to arbitrary destinations while having control of the contents of the request itself. SSRF vulnerabilities often arise from implementations where our web application needs to use third-party services.

Questions

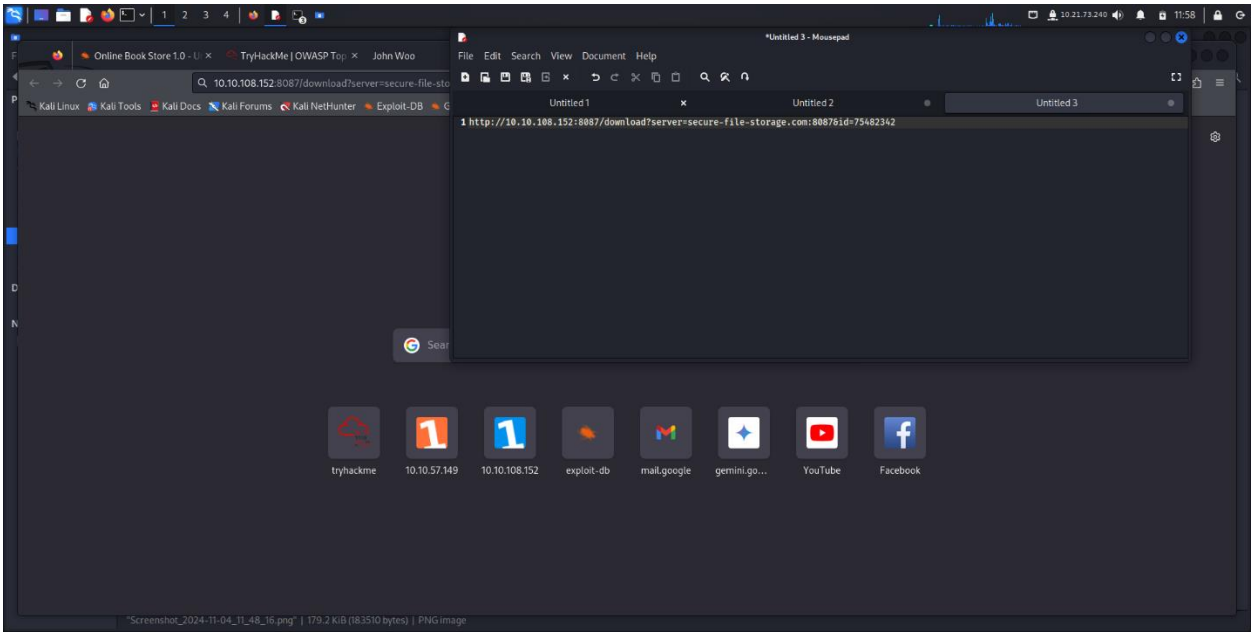


Quiz 2

I started of by loading the IP on the browser and found it was a portfolio for John containing his resume and works, I copied the link to the download button,

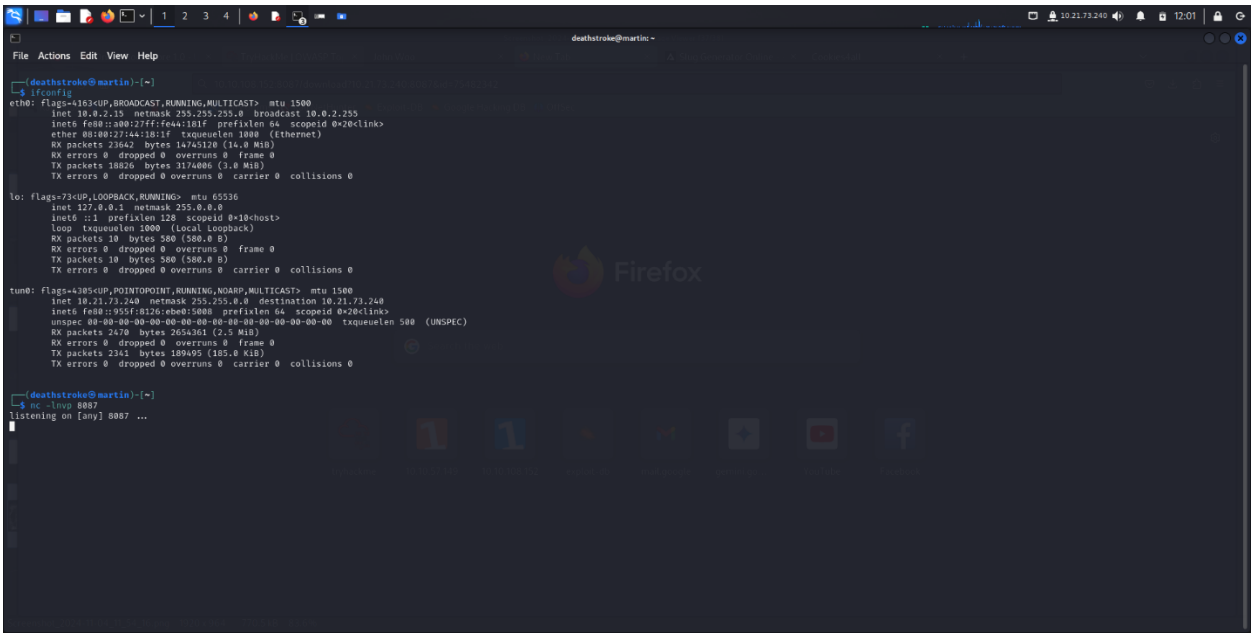


This how the link Looked like, when downloading one was being redirected to another sever, secure-file-storage.com.

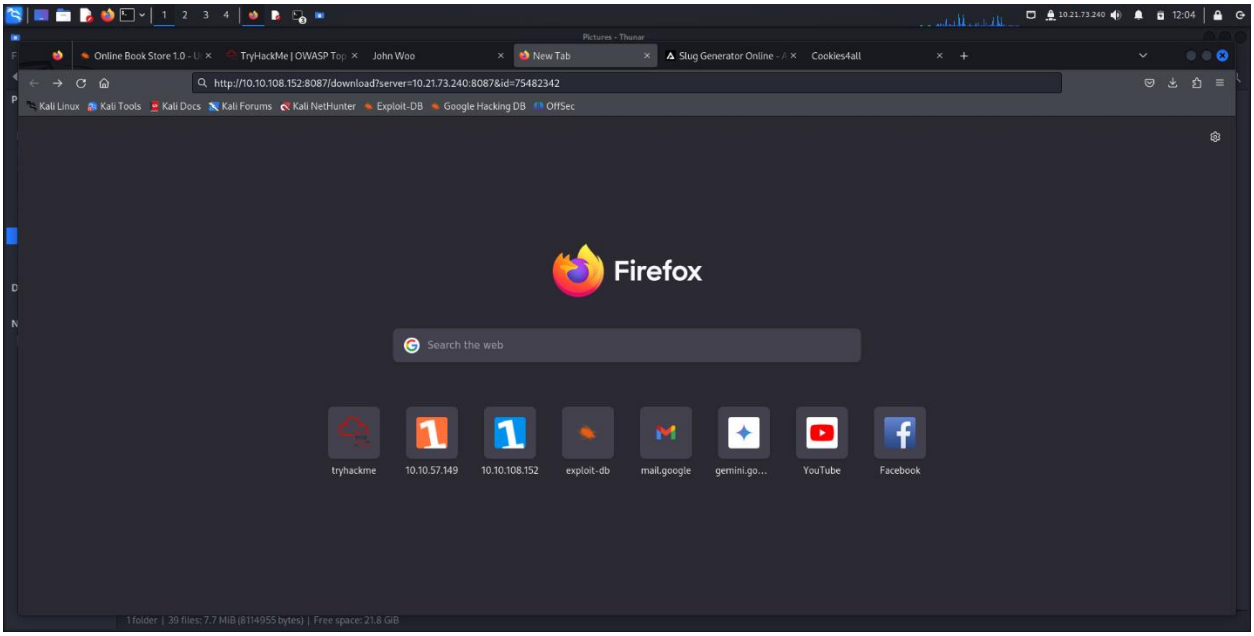


Quiz 3

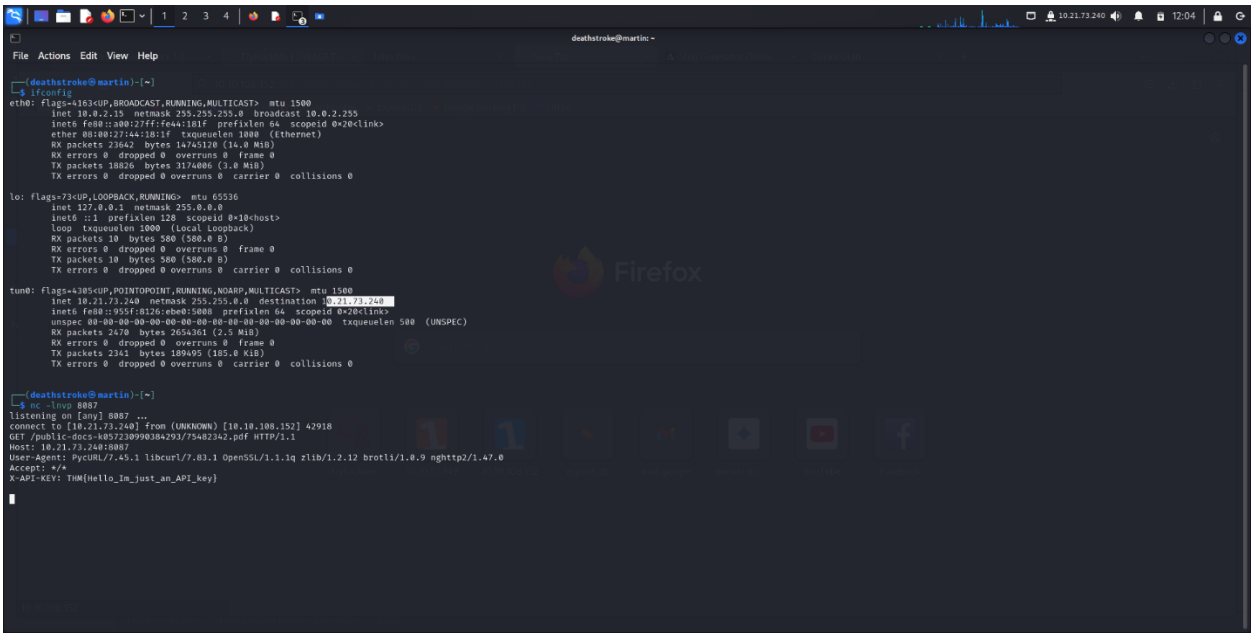
Used command ifconfig to get the IP address I was assigned after connecting to the vpn, and decided to set a netcat to listen at port 8087, so as to intercept the contents of the request,



I modified the Link of download button, to have my IP address instead the secure-file-storage,com,



And when I reloaded the browser I found reply on netcat listening, it had already intercepted the contents of the request, and viewed the flag.s



2.0 In conclusion

This room has been beneficial and insightful of the various security risk that a web application can experience.