TITLE – Try Hack Me
OWASP TOP 10 - 2021

Task 1: Introduction

The following report provides a comprehensive overview and write-up for each lab on TryHackMe that focuses on the OWASP Top 10 2021. The labs covered in this report are designed to address various security vulnerabilities, including

1. Broken Access Control
2. Cryptographic Failures
3. Injection
4. Insecure Design
5. Security Misconfiguration
6. Vulnerable and Outdated Components
7. Identification and Authentication Failures
8.
9. Software and Data Integrity Failures
   Security Logging & Monitoring Failures
10. Server-Side Request Forgery (SSRF)

Task 2: Accessing Machine

- ☐ Start Machine

☐ Connect to tryhackme network using OpenVPN or deploy the AttackBox in the browser.

Task 3:1. Broken Access Control

Websites have pages that are protected from regular visitors. For example, only the site's admin user should be able to access a page to manage other users. If a website visitor can access protected pages they are not meant to see, then the access controls are broken.

A regular visitor being able to access protected pages can lead to the following:

- Being able to view sensitive information from other users
- Accessing unauthorized functionality

For example, a vulnerability was found in 2019, where an attacker could get any single frame from a Youtube video marked as private. The researcher who found the vulnerability showed that he could ask for several frames and somewhat reconstruct the video.

Task 4 : Broken Access Control (IDOR Challenge)

Insecure Direct Object Reference

IDOR or Insecure Direct Object Reference 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.

For example, let's say we're logging into our bank account, and after correctly authenticating ourselves, we get taken to a URL like this https://bank.thm/account?id=111111. On that page, we can see all our important bank details, and a user would do whatever they need to do and move along their way, thinking nothing is wrong.

https://bank.thm/account?id=111111

Welcome jclarke

Savings: $0,22

Account #111111

Account Owner: Jason Clarke

There is, however, a potentially huge problem here, anyone may be able to change the *id* parameter to something else like *222222,* and if the site is incorrectly configured, then he would have access to someone else's bank information.

https://bank.thm/account?id=222222

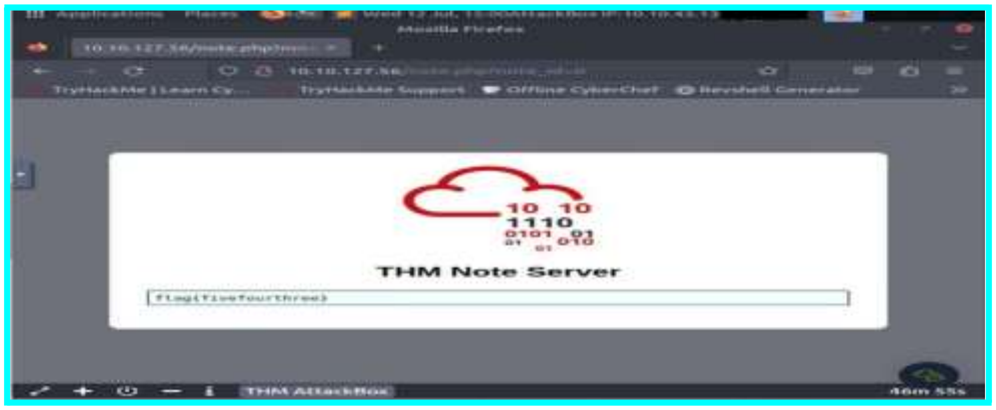
Welcome jclarke

Savings: $1337,00

Account #222222

Account Owner: Nick Perry

- ☐ First of all login
- ☐ Put id=0 (http://machine-ip/note.php?note_id=0) and you will get the flag:



THM Note Server

flag{fivefourthree}

Answer the questions below :

1. Read and understand how IDOR works.

A. No answer needed

2. Deploy the machine and go to http://MACHINE_IP - Login with the username noot and the password test1234.

A. No answer needed3. Look at other users' notes. What is the flag?

A. flag{fivefourthree}

Task 5 : 2. Cryptographic Failures

The Cryptographic Failures lab on TryHackMe is designed to showcase the vulnerabilities associated with improper cryptographic implementation in web applications. This lab focuses on demonstrating scenarios where cryptographic algorithms, protocols, or key management practices are flawed, leading to security weaknesses.

Objective:

The objective of this lab is to understand the impact of cryptographic failures and how they can be exploited by attackers. By simulating various cryptographic vulnerabilities, participants will gain hands-on experience in identifying and exploiting these weaknesses.

Take, for example, a secure email application:

- When you are accessing your email account using your browser, you want to be sure that the communications between you and the server are encrypted. That way, any eavesdropper trying to capture your network packets won't be able to recover the content of your email addresses.

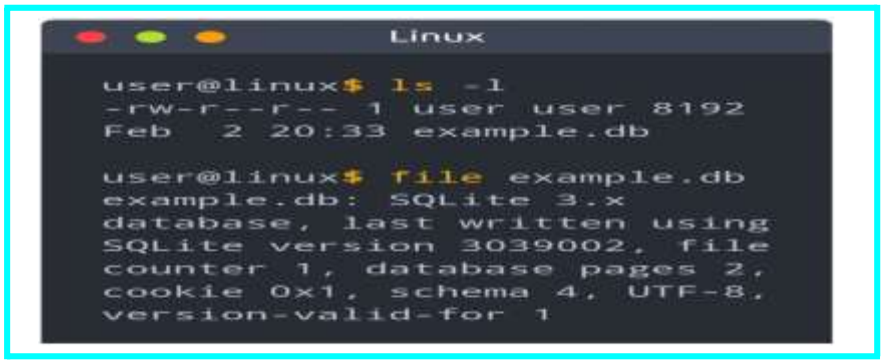
- Since your emails are stored in some server managed by your provider, it is also desirable that the email provider can't read their client's emails. To this end, your emails might also be encrypted when stored on the servers.

Task 6 : Cryptographic Failures(Supporting Material 1)

The most common way to store a large amount of data in a format easily accessible from many locations is in a database. This is perfect for something like a web application, as many users may interact with the website at any time. Database engines usually follow the Structured Query Language (SQL) syntax.

As mentioned previously, flat-file databases are stored as a file on the disk of a computer. Usually, this would not be a problem for a web app, but what happens if the database is stored underneath the root directory of the website (i.e. one of the files accessible to the user connecting to the website)? The most common (and simplest) format of a flat-file database is an SQLite database. These can be interacted with in most programming languages and have a dedicated client for querying them on the command line. This client is called sqlite3 and is installed on many Linux distributions by default.

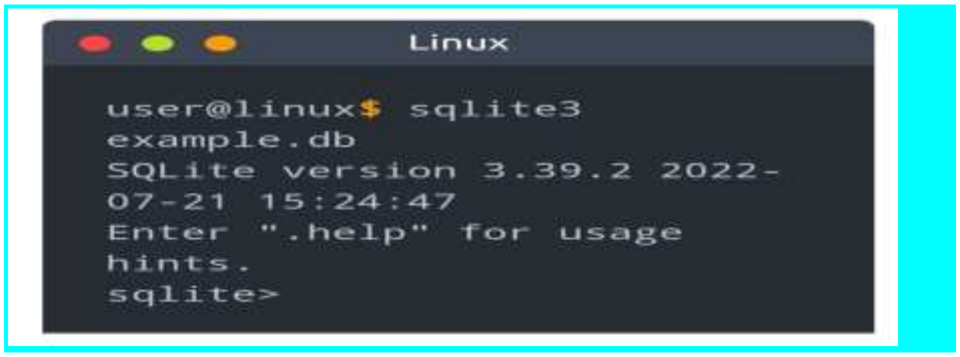
Let's suppose we have successfully managed to download a database:



```
user@linux$ ls -l
-rw-r--r-- 1 user user 8192
Feb  2 20:33 example.db

user@linux$ file example.db
example.db: SQLite 3.x
database, last written using
SQLite version 3039002, file
counter 1, database pages 2,
cookie 0x1, schema 4, UTF-8,
version-valid-for 1
```

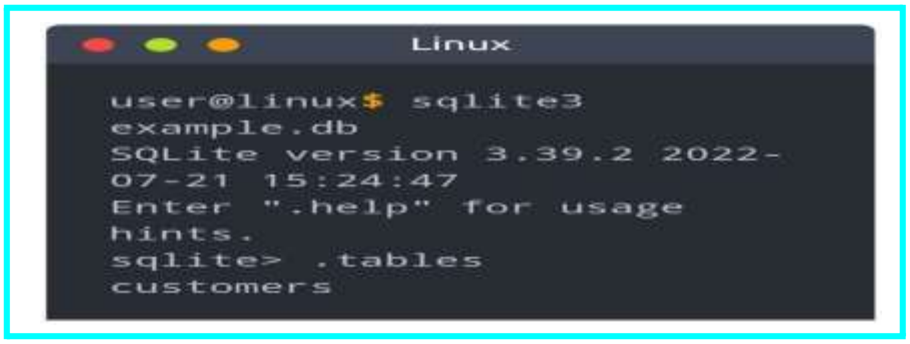
We can see that there is an SQLite database in the current folder.

To access it, we use sqlite3<database-name>



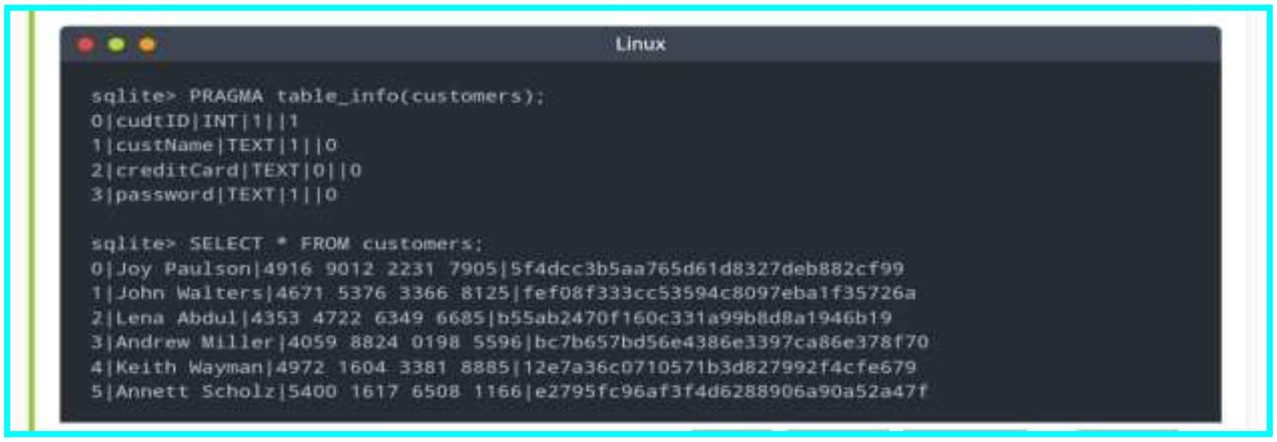
```
user@linux$ sqlite3
example.db
SQLite version 3.39.2 2022-
07-21 15:24:47
Enter ".help" for usage
hints.
sqlite>
```

From here, we can see the tables in the database by using the .tables command:



```
user@linux$ sqlite3
example.db
SQLite version 3.39.2 2022-
07-21 15:24:47
Enter ".help" for usage
hints.
sqlite> .tables
customers
```

At this point, we can dump all the data from the table, but we won't necessarily know what each column means unless we look at the table information. First, let's use PRAGMA table_info(customers); to see the table information. Then we'll use SELECT * FROM customers; to dump the information from the table:



```
Linux
sqlite> PRAGMA table_info(customers);
0|cudtID|INT|1||1
1|custName|TEXT|1||0
2|creditCard|TEXT|0||0
3|password|TEXT|1||0

sqlite> SELECT * FROM customers;
0|Joy Paulson|4916 9012 2231 7905|5f4dcc3b5aa765d61d8327deb882cf99
1|John Walters|4671 5376 3366 8125|fef08f333cc53594c8097eba1f35726a
2|Lena Abdul|4353 4722 6349 6685|b55ab2470f160c331a99b8d8a1946b19
3|Andrew Miller|4059 8824 0198 5596|bc7b657bd56e4386e3397ca86e378f70
4|Keith Wayman|4972 1604 3381 8885|12e7a36c0710571b3d827992f4cfe679
5|Annett Scholz|5400 1617 6508 1166|e2795fc96af3f4d6288906a90a52a47f
```

We can see from the table information that there are four columns: custID, custName, creditCard and password. You may notice that this matches up with the results. Take the first row:
0|Joy Paulson|4916 9012 2231 7905|5f4dcc3b5aa765d61d8327deb882cf99

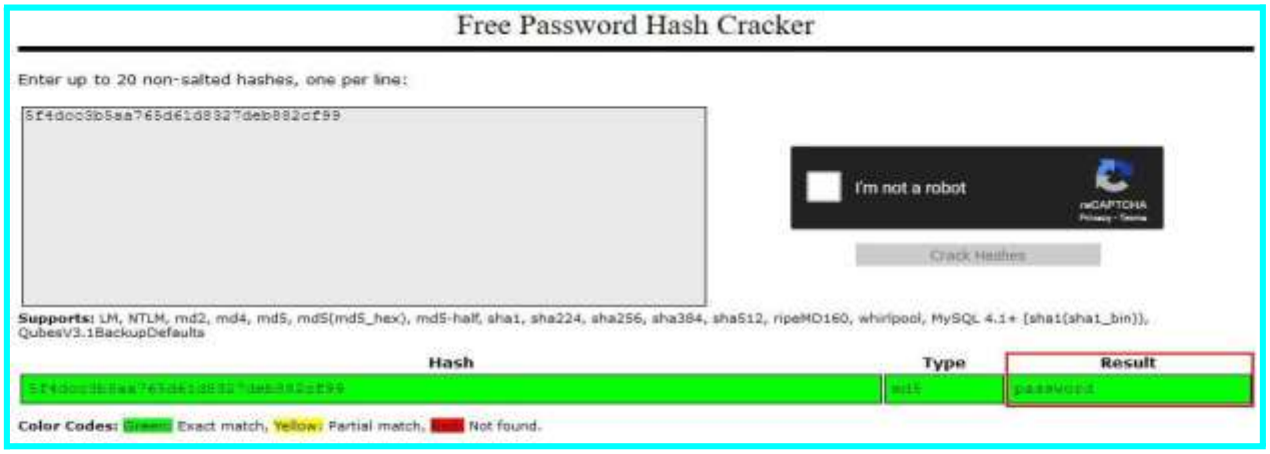
We have the custID (0), the custName (Joy Paulson), the creditCard (4916 9012 2231 7905) and a password hash (5f4dcc3b5aa765d61d8327deb882cf99).

In the next task, we'll look at cracking this hash.

Task 7 :Cryptographic Failures (Supporting Material 2)

We saw how to query an SQLite database for sensitive data in the previous task. We found a collection of password hashes, one for each user. In this task, we will briefly cover how to crack these.Instead, we will be using the online tool: Crackstation. This website is extremely good at cracking weak password hashes. For more complicated hashes, we would need more sophisticated tools; however, all of the crackable password hashes used in today's challenge are weak MD5 hashes, which Crackstation should handle very nicely.When we navigate to the website, we are met with the following interface:

Let's try pasting the password hash for Joy Paulson, which we found in the previous task (5f4dcc3b5aa765d61d8327deb882cf99). We solve the Captcha, then click the "Crack Hashes" button:



Free Password Hash Cracker

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

```
5f4dcc3b5aa765d61d8327deb882cf99
```

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
5f4dcc3b5aa765d61d8327deb882cf99	md5	password

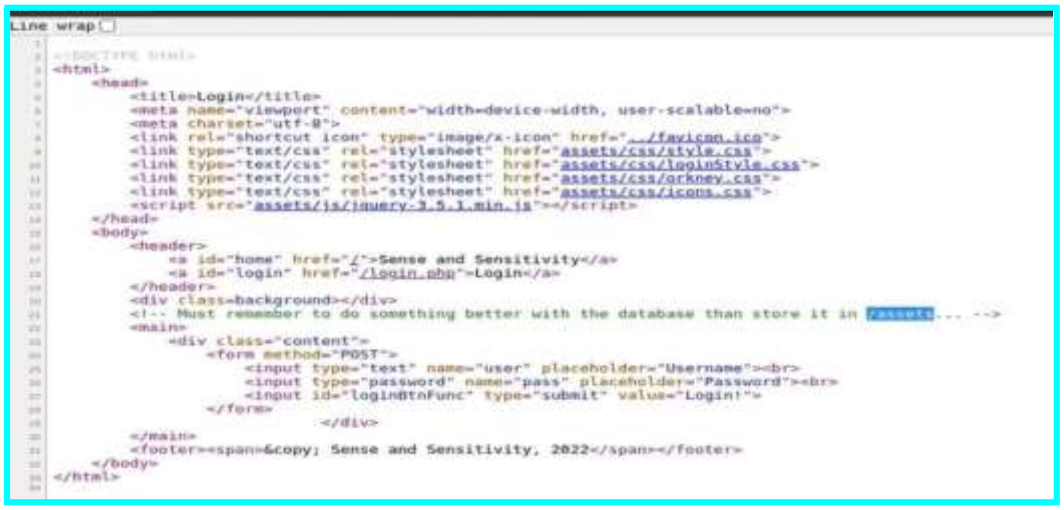
Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

We see that the hash was successfully broken, and the user's password was "password". How secure! It's worth noting that Crackstation works using a massive wordlist. If the password is not in the wordlist, then Crackstation will not be able to break the hash.

Task 8 : Cryptographic Failures (Challenge)

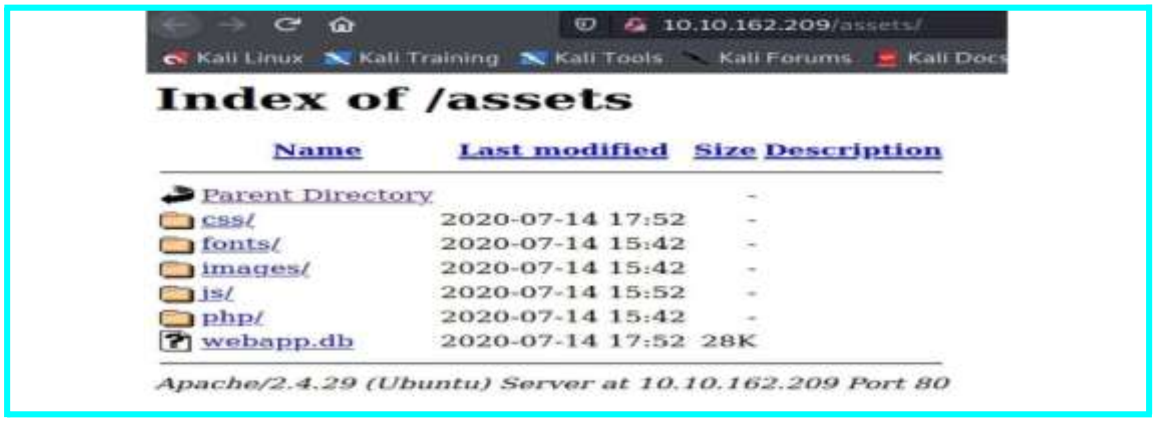
It's now time to put what you've learnt into practice! For this challenge, connect to the web application at http://MACHINE_IP:81/.

Go to the login page and view the source code to find the hidden directory.



```
Line wrap
<!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>
            <a id="home" href="/">Sense and Sensitivity</a>
            <a id="login" href="/login.php">Login</a>
        </header>
        <div class=background></div>
        <!-- Must remember to do something better with the database than store it in /assets... -->
        <main>
            <div class="content">
                <form method="POST">
                    <input type="text" name="user" placeholder="Username"><br>
                    <input type="password" name="pass" placeholder="Password"><br>
                    <input id="loginBtnFunc" type="submit" value="Login!">
                </form>
                        </div>
        </main>
        <footer><span>&copy; Sense and Sensitivity, 2022</span></footer>
    </body>
</html>
```

Download the webapp.db to examine it

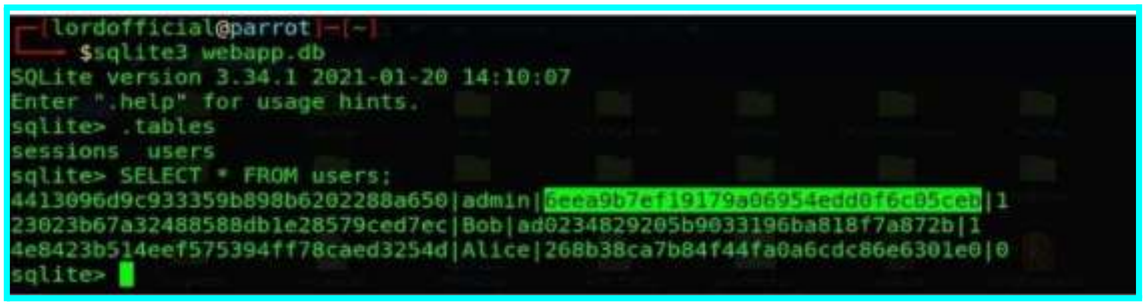


10.10.162.209/assets/

Index of /assets

```
Name              Last modified      Size  Description
Parent Directory                      -
css/              2020-07-14 17:52    -
fonts/            2020-07-14 15:42    -
images/           2020-07-14 15:42    -
js/               2020-07-14 15:52    -
php/              2020-07-14 15:42    -
webapp.db         2020-07-14 17:52   28K

Apache/2.4.29 (Ubuntu) Server at 10.10.162.209 Port 80
```

After Cracking the hash Now login as admin to get the flag.



```
┌─[lordofficial@parrot]─[~]
└──╼ $sqlite3 webapp.db
SQLite version 3.34.1 2021-01-20 14:10:07
Enter ".help" for usage hints.
sqlite> .tables
sessions  users
sqlite> SELECT * FROM users;
4413096d9c933359b898b6202288a650|admin|6eea9b7ef19179a06954edd0f6c05ceb|1
23023b67a32488588db1e28579ced7ec|Bob|ad0234829205b9033196ba818f7a872b|1
4e8423b514eef575394ff78caed3254d|Alice|268b38ca7b84f44fa0a6cdc86e6301e0|0
sqlite>
```

Answer the questions below :

Have a look around the web app. The developer has left themselves a note indicating that there is sensitive data in a specific directory.

1. What is the name of the mentioned directory?

A. /assets

2. Navigate to the directory you found in question one. What file stands out as being likely to contain sensitive data?

A. webapp.db

3. Use the supporting material to access the sensitive data. What is the password hash of the admin user?

A. 6eea9b7ef19179a06954edd0f6c05ceb

4. Crack the hash.

What is the admin's plaintext password?

A. qwertyuiop

5. Log in as the admin. What is the flag?

A. THM{Yzc2YjdkMjE5N2VjMzNhOTE3NjdiMjdl}

Task 9 : 3. Injection

The Injection lab on TryHackMe aims to demonstrate the vulnerabilities associated with code injection attacks in web applications. This lab focuses on showcasing scenarios where user-supplied input is not properly validated or sanitized, leading to the execution of unintended commands or SQL queries.

Objective: The objective of this lab is to understand the impact of injection attacks and how they can be exploited by attackers. By simulating various injection vulnerabilities, participants will gain hands-on experience in identifying and exploiting these weaknesses.

Conclusion: The Injection lab effectively demonstrates the consequences of code injection vulnerabilities in web applications. By exploiting these vulnerabilities, participants gain a deeper understanding of the impact of injection attacks and the importance of implementing proper input validation and sanitization measures to prevent unauthorized access and data leakage.

Task 10 : 3.1. Command Injection

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. MooCorp has started developing a web-based application for cow ASCII art with customisable text. While searching for ways to implement their app, they've come across the `cowsay` command in Linux, which does just that! Instead of coding a whole web application and the logic required to make cows talk in ASCII, they decide to write some simple code that calls the `cowsay` command from the operating system's console and sends back its contents to the website.`<?php`

```
  if (isset($_GET["mooing"])) {
      $mooing = $_GET["mooing"];
      $cow = 'default';
     if(isset($_GET["cow"]))
          $cow = $_GET["cow"];
 passthru("perl /usr/bin/cowsay -f $cow $mooing");
  }
```

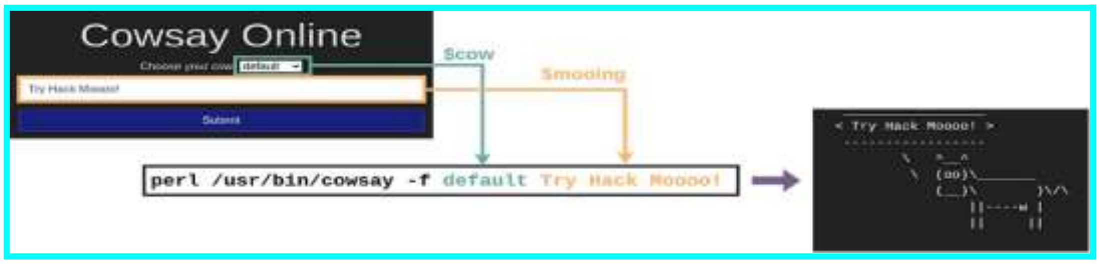
```
?>
```

In simple terms, the above snippet does the following:

1. Checking if the parameter “mooing” is set. If it is, the variable $mooing gets what was passed into the input field.

2.Checking if the parameter “cow” is set. If it is, the variable $cow gets what was passed through the parameter.

3.The program then executes the function passthru("perl /usr/bin/cowsay -f $cow $mooing");. The passthru function simply executes a command in the operating system's console and sends the output back to the user's browser. You can see that our command is formed by concatenating the $cow and $mooing variables at the end of it. Since we can manipulate those variables, we can try injecting additional commands by using simple tricks. If you want to, you can read the docs on passthru() on [PHP's website]() for more information on the function itself



```
perl /usr/bin/cowsay -f default Try Hack Moooo!
```

```
 _________________
< Try Hack Moooo! >
 -----------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

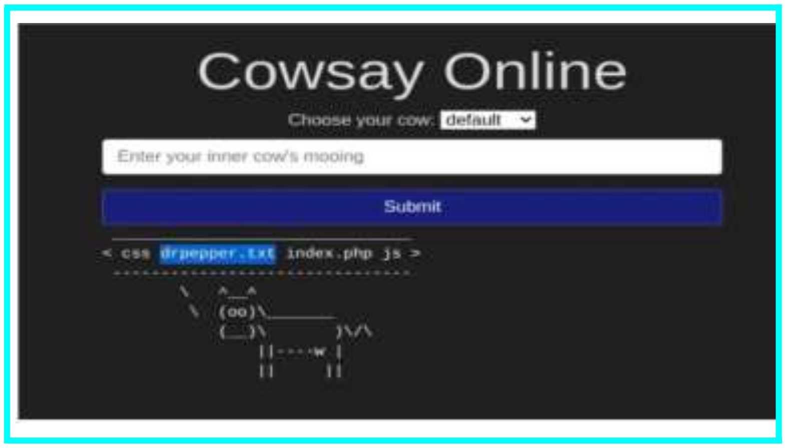
Exploiting Command Injection

To execute inline commands, you only need to enclose them in the following format `$(your_command_here)`. If the console detects an inline command, it will execute it first and then use the result as the parameter for the outer command. Look at the following example, which runs `whoami` as an inline command inside an `echo` command:

Since the application accepts any input from us, we can inject an inline command which will get executed and used as a parameter for cowsay. This will make our cow say whatever the command returns! In case you are not that familiar with Linux, here are some other commands you may want to try:

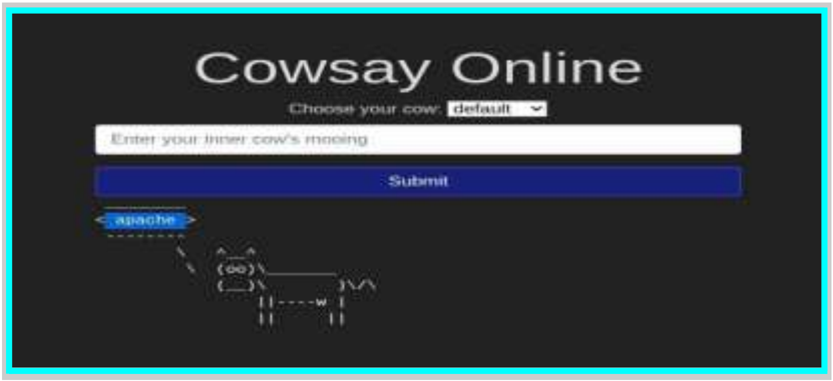
- whoami
- id
- ifconfig/ip addr
- uname -a
- ps -ef

Command Used : $(ls)



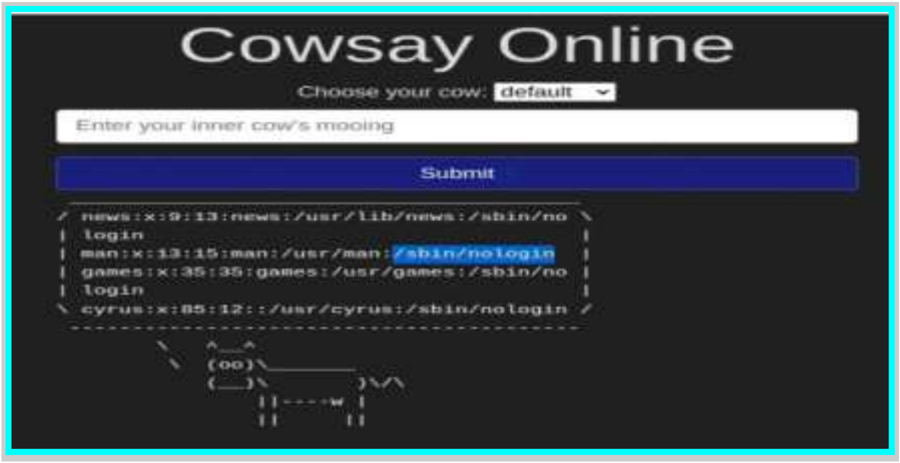
```
 _______________________________
< css drpepper.txt index.php js >
 -------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Command Used : $(whoami)



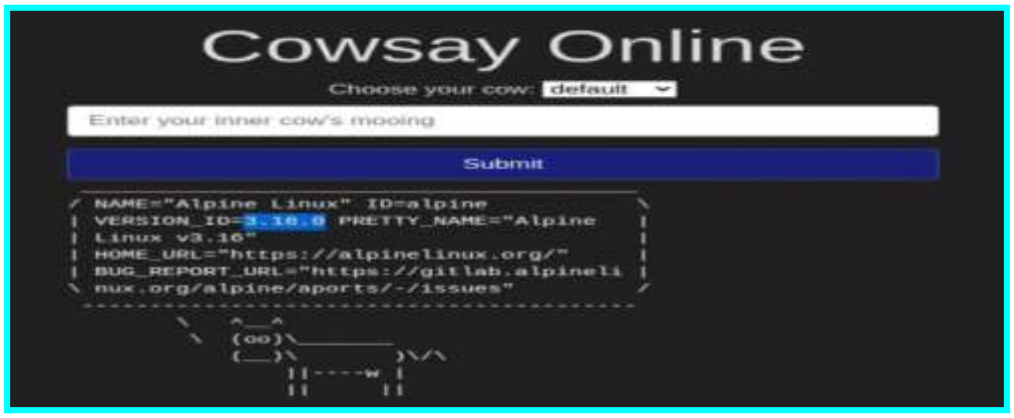
```
 ________
< apache >
 --------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Command Used : $(cat /etc/passwd | grep “usr”)



```
 ________________________________________
/ news:x:9:13:news:/usr/lib/news:/sbin/no \
| login                                   |
| man:x:13:15:man:/usr/man:/sbin/nologin  |
| games:x:35:35:games:/usr/games:/sbin/no |
| login                                   |
\ cyrus:x:85:12::/usr/cyrus:/sbin/nologin /
 -----------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Command Used : $(cat /etc/os-release



```
 ________________________________________
/ NAME="Alpine Linux" ID=alpine           \
| VERSION_ID=3.16.0 PRETTY_NAME="Alpine   |
| Linux v3.16"                            |
| HOME_URL="https://alpinelinux.org/"     |
| BUG_REPORT_URL="https://gitlab.alpineli |
\ nux.org/alpine/aports/-/issues"         /
 -----------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Answer the questions below :

1. What strange text file is in the website's root directory?
A. drpepper.txt

2. How many non-root/non-service/non-daemon users are there?
A. 0

3. What user is this app running as?
A. apache

4. What is the user's shell set as?
A. /sbin/nologin

5. What version of Alpine Linux is running?
A. 3.16.0

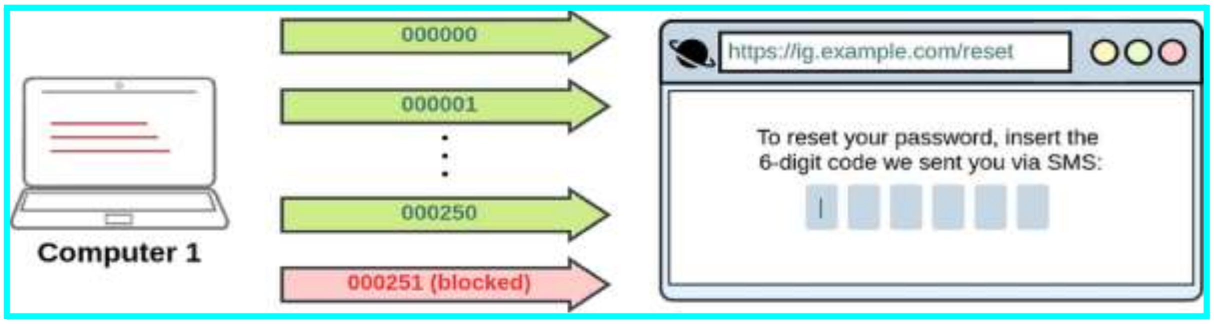
Task 11 : 4. Insecure Design

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 (or a part of it) is flawed from the start. A developer could, for example, disable the OTP validation in the development phases to quickly test the rest of the app without manually inputting a code at each login but forget to re-enable it when sending the application to production.

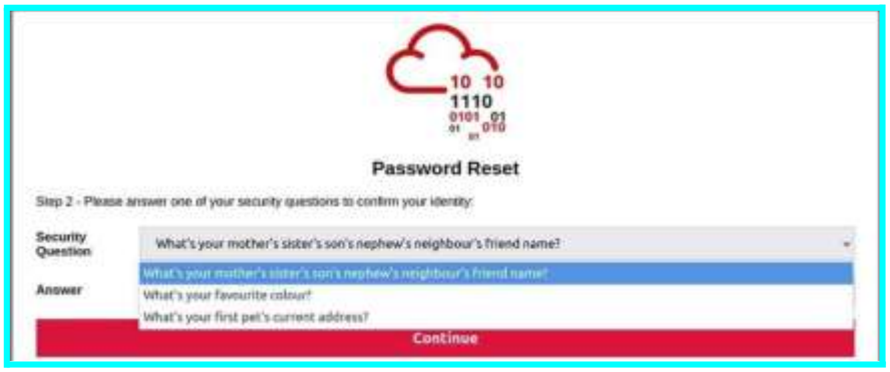
Insecure Password Resets

A good example of such vulnerabilities occurred on Instagram a while ago. Instagram allowed users to reset their forgotten passwords by sending them a 6-digit code to their mobile number via SMS for validation. If an attacker wanted to access a victim's account, he could try to brute-force the 6-digit code. As expected, this was not directly possible as Instagram had rate-limiting implemented so that after 250 attempts, the user would be blocked from trying further.


Computer 1
000000
000001
000250
000251 (blocked)
https://ig.example.com/reset
To reset your password, insert the 6-digit code we sent you via SMS:

If an attacker had several different IP addresses from where to send requests, he could now try 250 codes per IP. For a 6-digit code, you have a million possible codes, so an attacker would need 1000000/250 = 4000 IPs to cover all possible codes. This may sound like an insane amount of IPs to have, but cloud services make it easy to get them at a relatively small cost, making this attack feasible.

Practical Example :- Navigate to http://machine-ip: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?Now go to the password reset page of **joseph** and try to give the answer to the Security question. Make a wild guess just like id

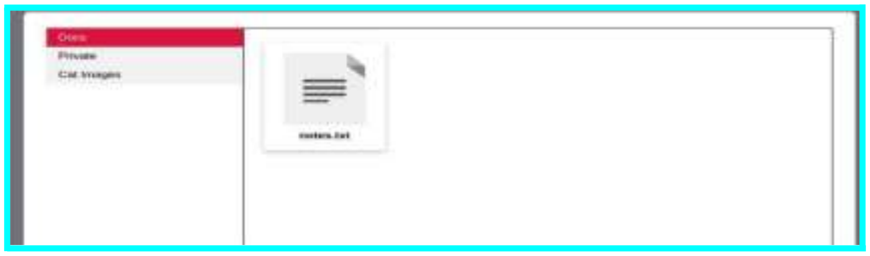


Step 2 - Please answer one of your security questions to confirm your identity:

Security Question: What's your mother's sister's son's nephew's neighbour's friend name?

What's your mother's sister's son's nephew's neighbour's friend name?
What's your favourite colour?
What's your first pet's current address?

And finally we got his password so, try to login with this new password.
We got successfully logged in & the only thing which is left is to open the flag.txt file to get the flag.



Answer the questions below :

1. Try to reset joseph's password. Keep in mind the method used by the site to validate if you are indeed joseph.
A. No answer needed

2. What is the value of the flag in Joseph's account?
A. THM{Not_3ven_c4tz_c0uld_sav3_U!}

Task 12 : 5. Security Misconfiguration

Security Misconfiguration

Security Misconfigurations are distinct from the other Top 10 vulnerabilities because they 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:

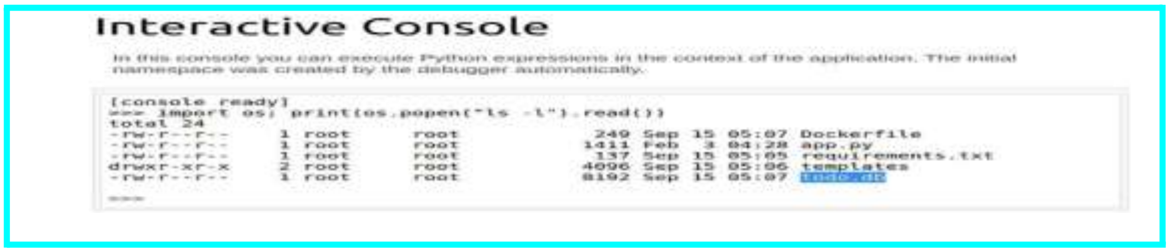
- Poorly configured permissions on cloud services, like S3 buckets.
- Having unnecessary features enabled, like services, pages, accounts or privileges.
- Default accounts with unchanged passwords.
- Error messages that are overly detailed and allow attackers to find out more about the system.

- Not using HTTP security headers.

This vulnerability can often lead to more vulnerabilities, such as default credentials giving you access to sensitive data, XML External Entities (XXE) or command injection on admin pages.For more info, look at the OWASP top 10 entry for Security Misconfiguration.

Debugging Interfaces

A common security misconfiguration concerns the exposure of debugging features in production software. Debugging features 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.One example of such a vulnerability was allegedly used when Patreon got hacked in 2015. Five days before Patreon was hacked, a security researcher reported to Patreon that he had found an open debug interface for a Werkzeug console. Werkzeug is a vital component in Python-based web applications as it provides an interface for web servers to execute the Python code. Werkzeug includes a debug console that can be accessed either via URL on `/console`, or it will also be presented to the user if an exception is raised by the application.



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.

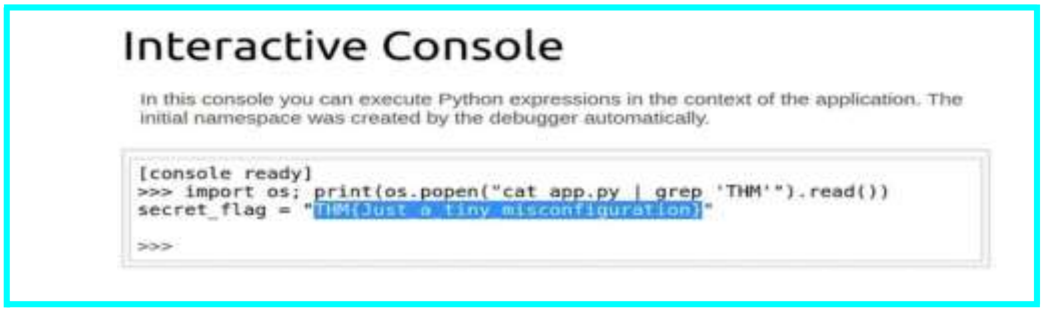
```
[console ready]
>>> import os; print(os.popen("ls -l").read())
total 24
-rw-r--r--    1 root      root           249 Sep 15 05:07 Dockerfile
-rw-r--r--    1 root      root          1411 Feb  3 04:28 app.py
-rw-r--r--    1 root      root           137 Sep 15 05:05 requirements.txt
drwxr-xr-x    2 root      root          4096 Sep 15 05:06 templates
-rw-r--r--    1 root      root          8192 Sep 15 05:07 todo.db

>>>
```

Practical example

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

Navigate to http://machine-ip:86 and try to exploit the security misconfiguration to read the application's source code.

Command Used : import os; print(os.popen("ls -l").read())



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("cat app.py | grep 'THM'").read())
secret_flag = "THM{Just_a_tiny_misconfiguration}"

>>>
```

Answer the questions below :

1. Navigate to http://MACHINE_IP:86/console to access the Werkzeug console.

➔ No answer Needed

2. 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())`

3. What is the database file name (the one with the .db extension) in the current directory?

   ➔ todo.db

4. 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}

Task 13 : 6. Vulnerable and Outdated Components

Vulnerable and Outdated Components

Vulnerable and outdated components, deliberately integrated within the virtual machines or systems provided on TryHackMe, serve as a common aspect to emphasize the significance of maintaining up-to-date software and to demonstrate the potential risks associated with utilizing obsolete or vulnerable versions.

Occasionally, you may find that the company/entity you're pen-testing is using a program with a well-known vulnerability.

For example, let's say that a company hasn't updated their version of WordPress for a few years, and using a tool such as WPScan, you find that it's version 4.6. Some quick research will reveal that WordPress 4.6 is vulnerable to an unauthenticated remote code execution(RCE) exploit, and even better, you can find an exploit already made on Exploit-DB.

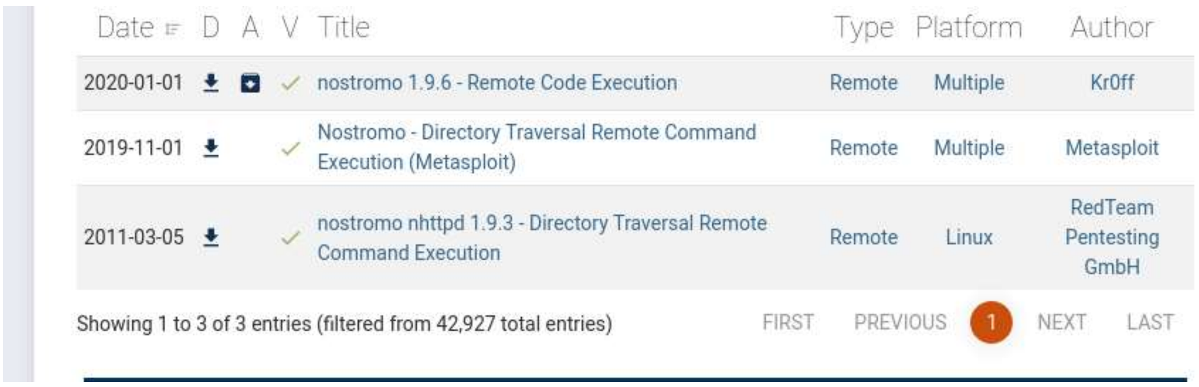
As you can see, this would be quite devastating because it requires very little work on the attacker's part. Since the vulnerability is already well known, someone else has likely made an exploit for the vulnerability already. The situation worsens when you realise that it's really easy for this to happen. If a company misses a single update for a program they use, it could be vulnerable to any number of attacks.

Task 14 : 6.1. Vulnerable and Outdated Components – Exploit

Vulnerable and Outdated Components – Exploit

Since this is about known vulnerabilities, most of the work has already been done for us. Our main job is to find out the information of the software and research it until we can find an exploit. Let's go through that with an example web application.

Upon discovering that the server is running the Nostromo web server with version 1.96, we can leverage resources such as Exploit-DB to search for potential exploits tailored to this specific version.



Date	D	A	V	Title	Type	Platform	Author
2020-01-01	⬇	▣	✓	nostromo 1.9.6 - Remote Code Execution	Remote	Multiple	Kr0ff
2019-11-01	⬇		✓	Nostromo - Directory Traversal Remote Command Execution (Metasploit)	Remote	Multiple	Metasploit
2011-03-05	⬇		✓	nostromo nhttpd 1.9.3 - Directory Traversal Remote Command Execution	Remote	Linux	RedTeam Pentesting GmbH

Showing 1 to 3 of 3 entries (filtered from 42,927 total entries)

the top result happens to be an exploit script. Let's download it and try to get code execution.

```
Linux
user@linux$ python 47837.py
Traceback (most recent call last):
  File "47837.py", line 10, in <module>
    cve2019_16278.py
NameError: name 'cve2019_16278' is not defined
```

Upon attempting to run the downloaded exploit script, an error was encountered due to an undefined name 'cve2019_16278'. Fortunately, in this case, the error occurred due to an uncommented line, making it a straightforward fix to address the issue. After addressing the necessary modifications, let's proceed to run the program again.

```
Linux
user@linux$ python2 47837.py 127.0.0.1 80 id
```

```
                              _____-2019-16278
      _____  _______    ______   _____\    \
  _____\    \_\      | |      | /    / |    |
 /     /|     ||     /  /     /|/    /  /___/|
/     / /____/||\    \  \    |/|    |__ |___|/
|     | |____|/ \ \    \ |    | |       \
|     |  _____   \|     \|    | |     __/ __
|\     \|\    \   |\         /| |\    \  /  \
| \_____\|    |   | \_______/ | | \____\/    |
| |     /____/|    \ |     | /  | |    |____/|
 \|_____|    ||     \|_____|/    \|____|   | |
        |____|/                        |___|/




HTTP/1.1 200 OK
Date: Fri, 03 Feb 2023 04:58:34 GMT
Server: nostromo 1.9.6
Connection: close

uid=1001(_nostromo) gid=1001(_nostromo) groups=1001(_nostromo)
```

After successfully achieving Remote Code Execution (RCE), it is important to note that most exploit scripts provide clear instructions regarding the required arguments, minimizing the need to delve into extensive code analysis.

However, it is crucial to acknowledge that not all scenarios will be as straightforward. Sometimes, additional investigation, such as examining HTML source code or making educated guesses, may be necessary to identify vulnerabilities. Nevertheless, for well-known vulnerabilities, it is typically possible to determine the application's version through research.

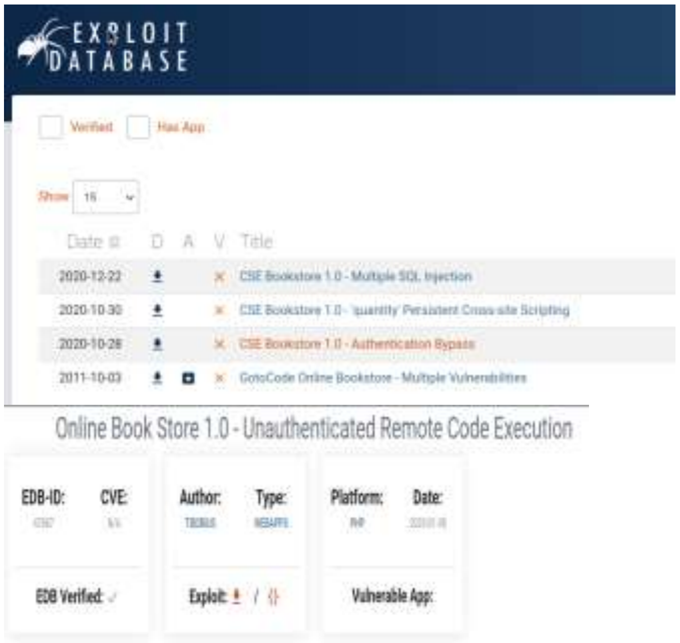
This aspect of penetration testing aligns with the convenience of leveraging existing work in the OWASP Top 10.

Task 15 : 6.2. Vulnerable and Outdated Components – Lab

Vulnerable and Outdated Components – Lab

Navigate to http://MACHINE_IP:84 where you'll find a vulnerable application.
This task is about exploiting outdated components. Open Firefox then head to the given site.

Looking at the site we learn that this is a bookstore app. after that go to the Exploit database site and search for the keywords bookstore .



Date	D	A	V	Title
2020-12-22	⬇		✗	CSE Bookstore 1.0 - Multiple SQL Injection
2020-10-30	⬇		✗	CSE Bookstore 1.0 - 'quantity' Persistent Cross-site Scripting
2020-10-28	⬇		✗	CSE Bookstore 1.0 - Authentication Bypass
2011-10-03	⬇	■	✗	GotoCode Online Bookstore - Multiple Vulnerabilities

Online Book Store 1.0 - Unauthenticated Remote Code Execution

EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
[illegible]	N/A	[illegible]	WEBAPPS	PHP	[illegible]

EDB Verified: ✓ Exploit: ⬇ / {} Vulnerable App:

Download the exploit (47887.py) and use the command:

python3 47887.py [insert URL] and type “y” to launch the shell.

Use the command [cat /opt/flag.txt] to reveal the flag.

The key to this is you already know what file you are looking for /opt/flag.txt

as we know that to open it we have to use [cat].

Answer the questions below :

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

Ans. THM{But_1ts_n0t_my_f4ult!}

Task 16 : 7. Identification and Authentication Failures

Identification and Authentication Failures

Authentication and session management constitute core components of modern web applications. 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. A user would enter these credentials, and the server would verify them. The server would then provide the users' browser with a session cookie if they are correct. A session cookie is 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.

If an attacker is able to find flaws in an authentication mechanism, they might successfully gain access to other users' accounts. This would allow the attacker to access sensitive data (depending on the purpose of the application). 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:** 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.

There can be various mitigation for broken authentication mechanisms depending on the exact flaw:

- To avoid password-guessing attacks, ensure the application enforces a strong password policy.
- To 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.
- Implement Multi-Factor Authentication. If a user has multiple authentication methods, for example, using a username and password and receiving a code on their mobile device, it would be difficult for an attacker to get both the password and the code to access the account.

Task 17 : 7.1. Identification and Authentication Failures Practical

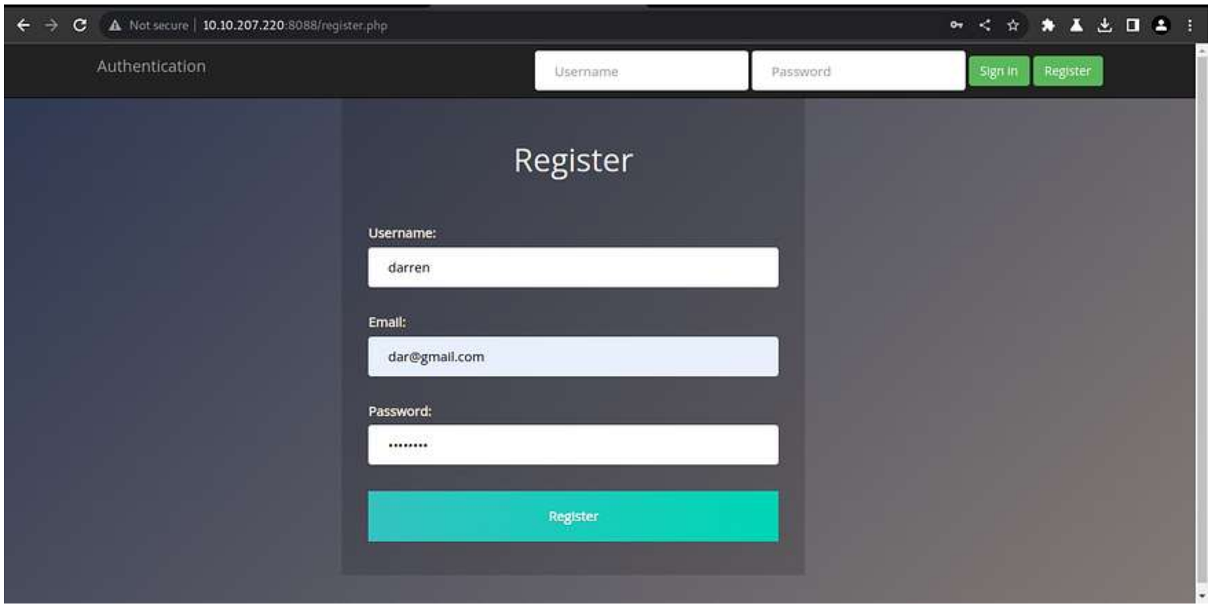
Identification and Authentication Failures Practical

In this example, we will examine a logic flaw within the authentication mechanism that arises due to a developer's oversight. Oftentimes, developers may neglect to properly sanitize user input, such as the username and password, within their application's code, making them susceptible to attacks like SQL injection. However, in this instance, we will concentrate on a vulnerability resulting from a developer's mistake that allows for the re-registration of an existing user.

Let's understand this with the help of an example, say there is an existing user with the name admin, and we want access to their account, so what we can do is try to re-register that username but with slight modification. We will enter " admin" without the quotes (notice the space at the start). Now when you enter that in the username field and enter other required information like email id or password and submit that data, it will register a new user, but that user will have the same right as the admin account. That new user will also be able to see all the content presented under the user admin.
To see this in action, go to http://MACHINE_IP:8088 and try to register with darren as your username. You'll see that the user already exists, so try to register " darren" instead, and you'll see that you are now logged in and can see the content present only in darren's account, which in our case, is the flag that you need to retrieve.

After going to the given site we need to register as Darren
But as a space in front of the name : “ darren”

If we try to register as :"darren" it will give error.
After that we tried to login using the credentials to find the flag



Not secure | 10.10.207.220:8088/register.php

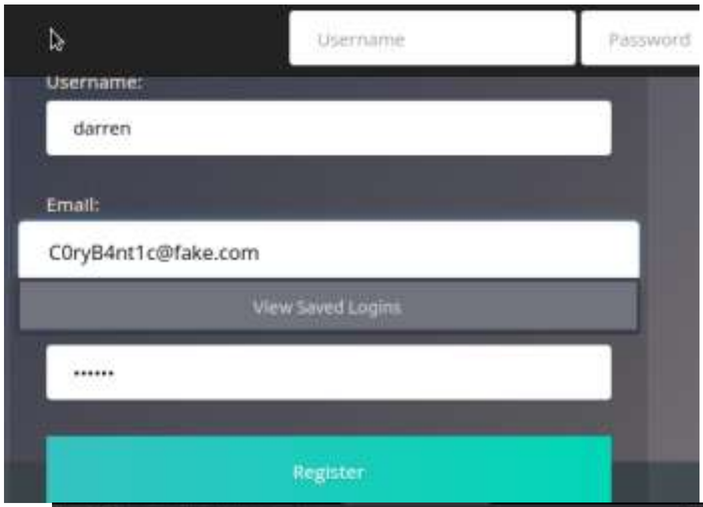
Authentication

Register

Username: darren

Email: dar@gmail.com

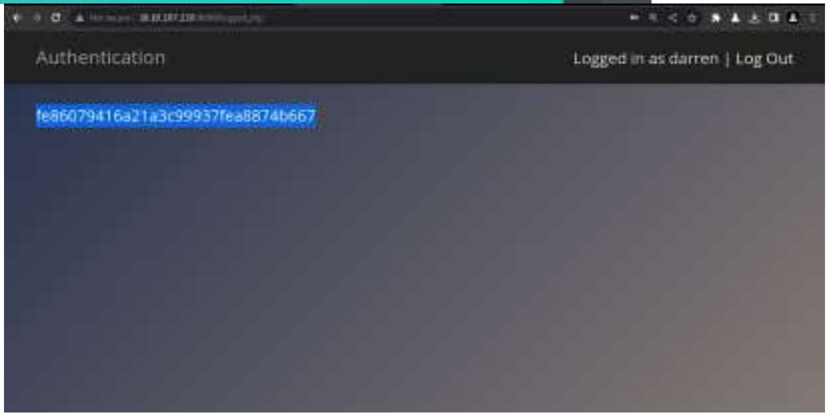
Password: ••••••••



Username: darren

Email: C0ryB4nt1c@fake.com

••••••



Authentication

Logged in as darren | Log Out

```
fe86079416a21a3c99937fea8874b667
```

Answer the questions below :

1. What is the flag that you found in darren's account?

ANS. fe86079416a21a3c99937fea8874b667

2. Now try to do the same trick and see if you can log in as **arthur.**
Ans. Correct answer

3. What is the flag that you found in arthur's account?
Ans. d9ac0f7db4fda460ac3edeb75d75e16e

Task 18 : 8. Software and Data Integrity Failures

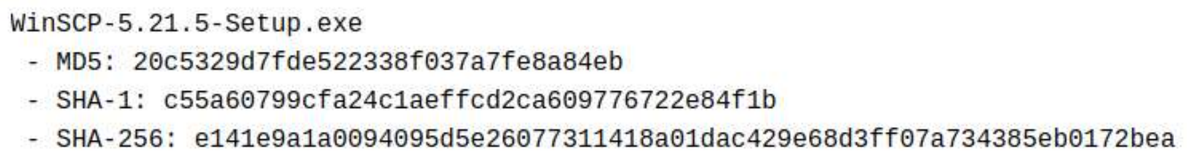
Software and Data Integrity Failures

What is Integrity?

When talking about integrity, we refer to the capacity we have to ascertain that a piece of data remains unmodified. Integrity is essential in cybersecurity as we care about maintaining important data free from unwanted or malicious modifications. For example, say you are downloading the latest installer for an application. How can you be sure that while downloading it, it wasn't modified in transit or somehow got damaged by a transmission error?

To overcome this problem, you will often see a **hash** sent alongside the file so that you can prove that the file you downloaded kept its integrity and wasn't modified in transit. A hash or digest is simply a number that results from applying a specific algorithm over a piece of data. When reading about hashing algorithms, you will often read about MD5, SHA1, SHA256 or many others available.

Let's take WinSCP as an example to understand better how we can use hashes to check a file's integrity. If you go to their Sourceforge repository, you'll see that for each file available to download, there are some hashes published along:



```
WinSCP-5.21.5-Setup.exe
 - MD5: 20c5329d7fde522338f037a7fe8a84eb
 - SHA-1: c55a60799cfa24c1aeffcd2ca609776722e84f1b
 - SHA-256: e141e9a1a0094095d5e26077311418a01dac429e68d3ff07a734385eb0172bea
```

These hashes were precalculated by the creators of WinSCP so that you can check the file's integrity after downloading. If we download theWinSCP-5.21.5-Setup.exe file,

we can recalculate the hashes and compare them against the ones published in Sourceforge. To calculate the different hashes in Linux, we can use the following commands:

```
AttackBox
user@attackbox$ md5sum WinSCP-5.21.5-Setup.exe
20c5329d7fde522338f037a7fe8a84eb  WinSCP-5.21.5-Setup.exe

user@attackbox$ sha1sum WinSCP-5.21.5-Setup.exe
c55a60799cfa24c1aeffcd2ca609776722e84f1b  WinSCP-5.21.5-Setup.exe

user@attackbox$ sha256sum WinSCP-5.21.5-Setup.exe
e141e9a1a0094095d5e26077311418a01dac429e68d3ff07a734385eb0172bea  WinSCP-5.21.5-
Setup.exe
```

Since we got the same hashes, we can safely conclude that the file we downloaded is an exact copy of the one on the website.

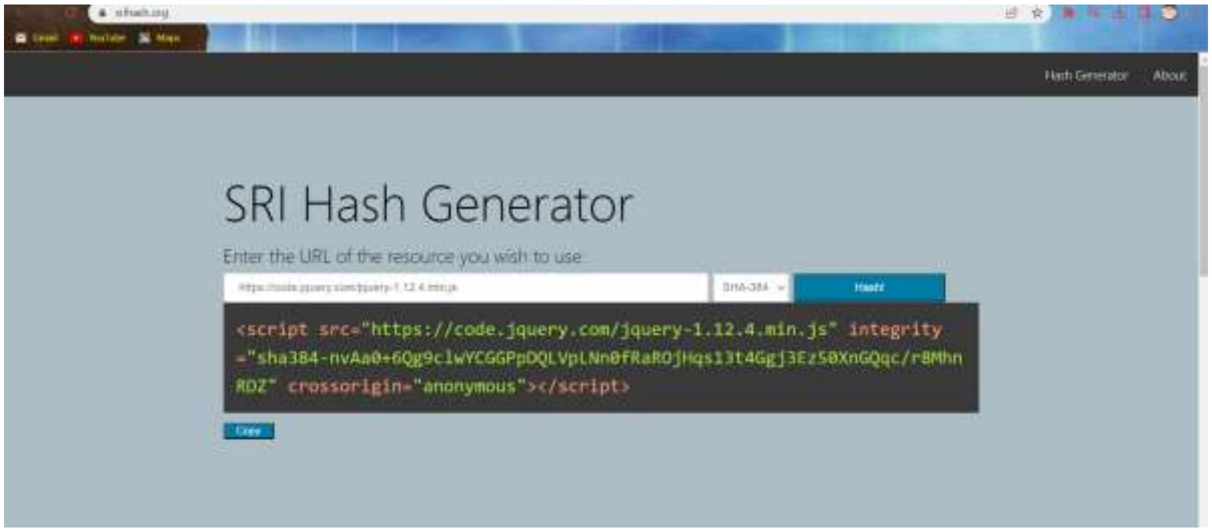
Software and Data Integrity Failures

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. There are mainly two types of vulnerabilities in this category:

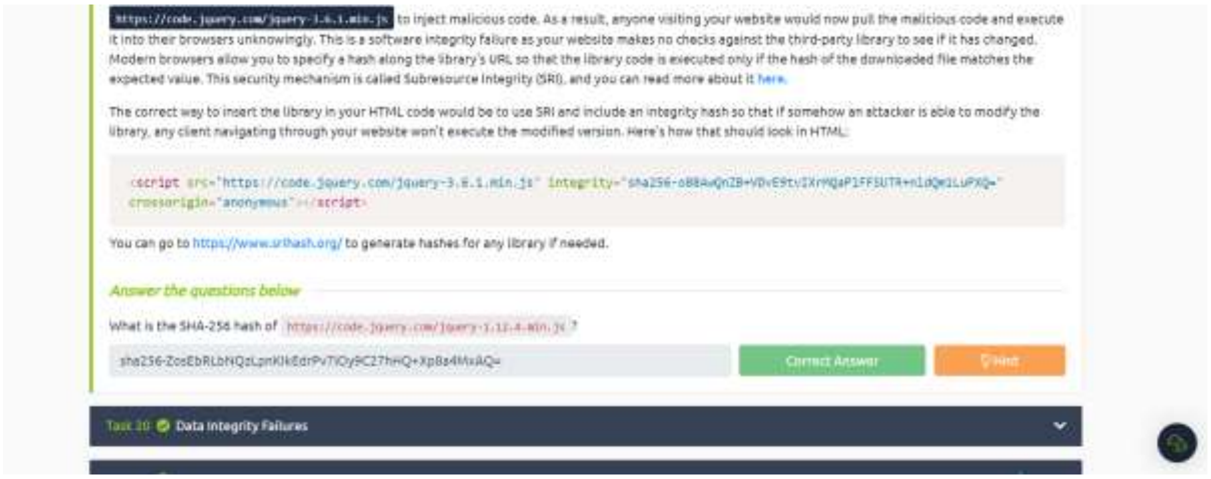
- Software Integrity Failures
- Data Integrity Failures

Task 19: Software Integrity Failures

Software integrity failures refer to instances where the security, reliability, or correctness of software systems is compromised. These failures can result in various negative consequences, such as unauthorized access, data breaches, system crashes, or the introduction of malicious code. To prevent software integrity failures, organizations should adopt secure coding practices, conduct regular security testing and vulnerability assessments, implement strong access controls, and stay up-to-date with security patches and updates. User education and the use of intrusion detection systems can also help improve software integrity.



```
<script src="https://code.jquery.com/jquery-1.12.4.min.js" integrity
="sha384-nvAa0+6Qg9clwYCGGPpDQLVpLNn0fRaROjHqs13t4Ggj3Ez50XnGQqc/r8Mhn
RDZ" crossorigin="anonymous"></script>
```



`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+VDvE9tvIXrMQaPlFFSUTR+nldQm1LuPXQ="
crossorigin="anonymous"></script>
```

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

Answer the questions below

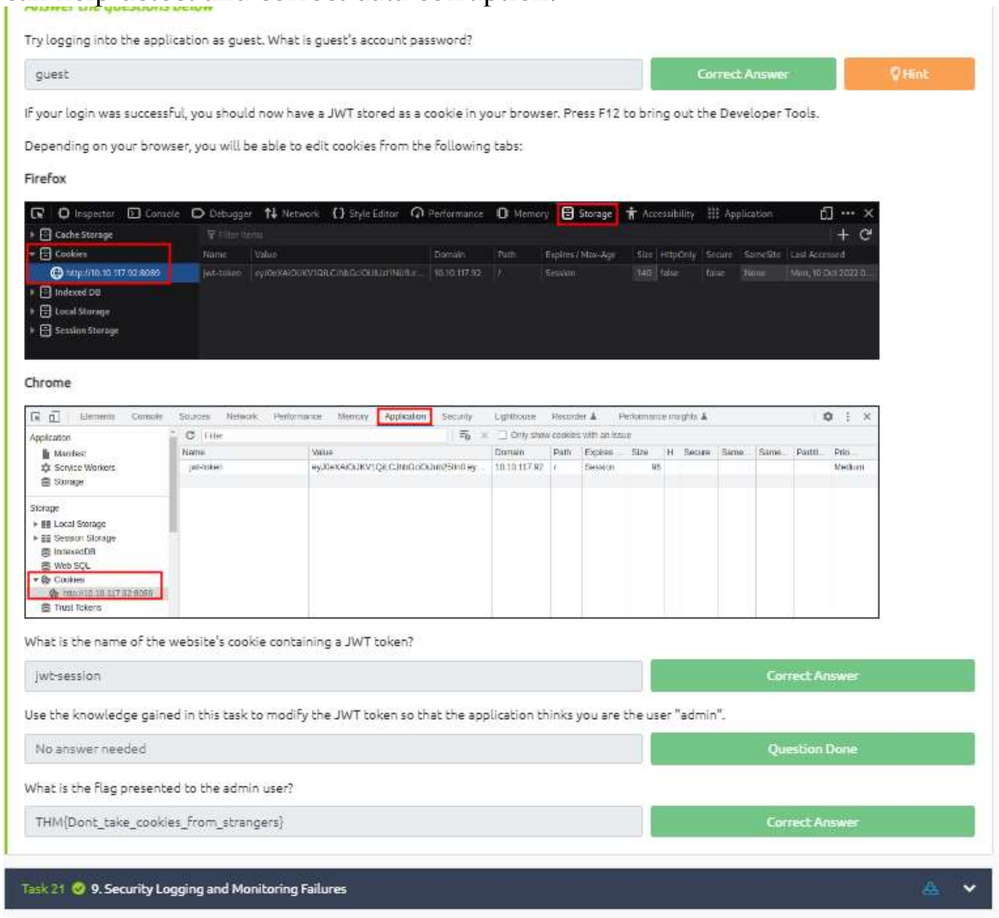
What is the SHA-256 hash of `https://code.jquery.com/jquery-1.12.4.min.js`?

sha256-ZosEbRLbNQzLpnKIkEdrPv7lOy9C27hHQ+Xp8a4MxAQ=

Task 20: Data Integrity Failures

Data integrity failures refer to instances where the accuracy, consistency, or reliability of data is compromised. These failures can occur due to various factors, such as human errors, software bugs, hardware malfunctions, or malicious activities. Data integrity failures can lead to incorrect or incomplete data, data corruption, or the loss of data altogether.
Here are some common examples of data integrity failures:
1. Data Corruption: Data corruption can occur due to various reasons, including software or hardware malfunctions, power outages, or network failures. It can result in data becoming

unreadable, inaccurate, or inconsistent.
2. Human Errors: Human mistakes, such as entering incorrect data, making formatting errors, or accidentally deleting or modifying data, can compromise data integrity. These errors can occur during data entry, data processing, or data transfer.
3. Software Bugs: Software bugs or programming errors can introduce flaws in data processing or storage mechanisms, leading to data integrity issues. These bugs may cause data to be modified, lost, or incorrectly interpreted.
4. Malicious Activities: Intentional actions by attackers or malicious insiders can compromise data integrity. This can include unauthorized access, data tampering, or injection of malicious code that alters or destroys data.
5. System Failures: Hardware or software failures, such as disk failures, memory corruption, or operating system crashes, can result in data integrity failures. These failures can lead to data loss or corruption.
6. Data Transfer Errors: During data transmission or migration processes, errors can occur that compromise data integrity. Network issues, data corruption during transit, or incomplete data transfers can all contribute to integrity failures.
7. Inadequate Backup and Recovery: Insufficient or ineffective data backup and recovery processes can result in data loss or incomplete restoration, impacting data integrity.
To mitigate data integrity failures, organizations should implement robust data management practices, including regular backups, data validation checks, and error detection mechanisms. Data encryption and access controls can help protect data from unauthorized modifications. Additionally, employing data integrity validation techniques, such as checksums or hash functions, can help detect and correct data corruption.



Try logging into the application as guest. What is guest's account password?

guest — Correct Answer — Hint

If your login was successful, you should now have a JWT stored as a cookie in your browser. Press F12 to bring out the Developer Tools.

Depending on your browser, you will be able to edit cookies from the following tabs:

Firefox

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
jwt-token	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.e...	10.10.117.92	/	Session	140	false	false	None	Mon, 10 Oct 2022 0...

Chrome

Name	Value	Domain	Path	Expires ...	Size	H	Secure	Same...	Same...	Partit...	Prio...
jwt-token	eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.ey...	10.10.117.92	/	Session	95						Medium

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

jwt-session — Correct Answer

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 — Question Done

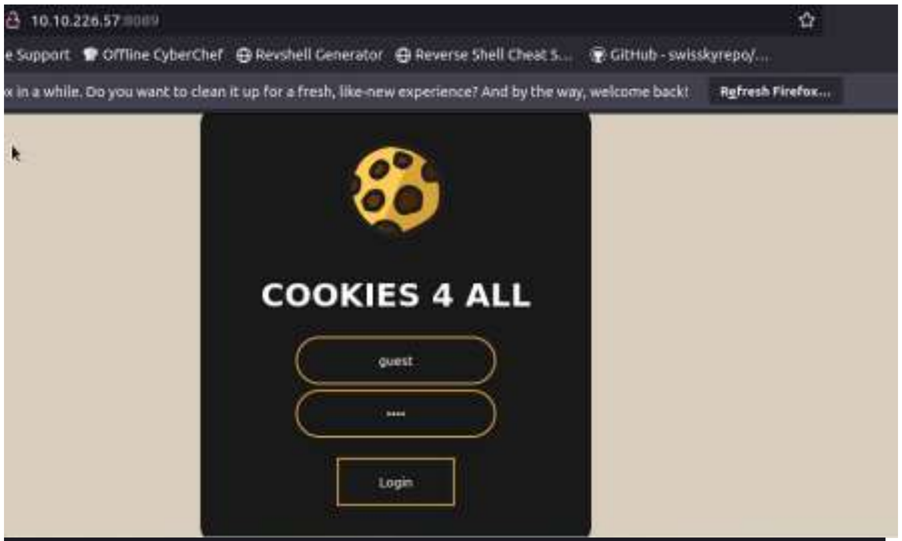
What is the flag presented to the admin user?

THM{Dont_take_cookies_from_strangers} — Correct Answer

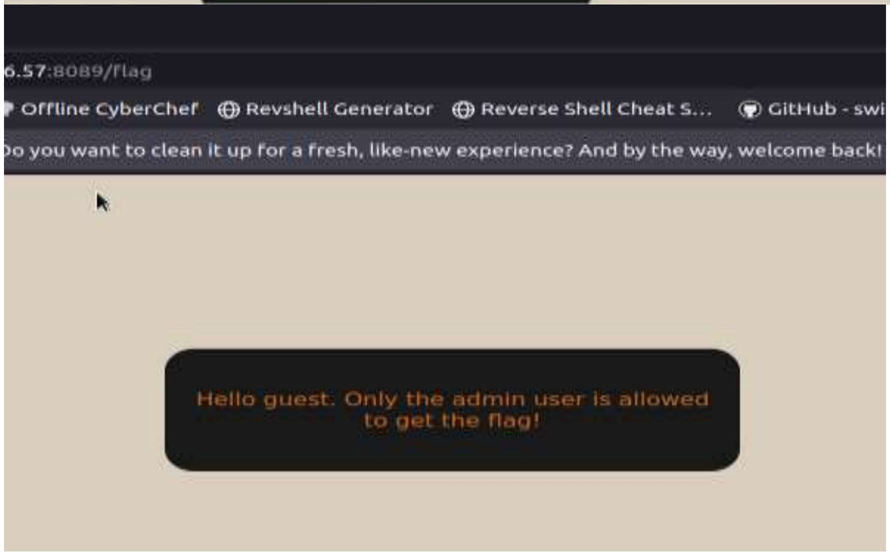
Task 21 9. Security Logging and Monitoring Failures

Task 21: Security Logging and Monitoring Failures

Security logging and monitoring failures refer to instances where the systems and processes designed to capture, store, and analyze security-related logs and events are compromised or ineffective. These failures can result in a lack of visibility into security incidents, delayed detection of threats, or the inability to effectively investigate and respond to security events.
Here are some examples of security logging and monitoring failures:
1. Inadequate Log Collection: Failure to collect logs from critical systems or applications can limit the ability to detect and investigate security incidents. It may occur due to misconfigurations, insufficient logging policies, or overlooked systems.
2. Insufficient Log Retention: Inadequate retention periods for security logs can result in the loss of valuable data needed for incident investigation, forensic analysis, or compliance requirements. Logs may be overwritten or purged before they are adequately analyzed.
3. Lack of Log Integrity: Failure to ensure the integrity of logs can compromise their reliability and trustworthiness. If logs can be tampered with or modified without detection, it becomes difficult to rely on them for accurate incident investigation or compliance purposes.
4. Ineffective Log Analysis: Failing to implement proper log analysis techniques and tools can lead to missed security events or delayed detection of threats. Without effective analysis, patterns, anomalies, or indicators of compromise may go unnoticed.
5. Inadequate Alerting and Notification: Insufficient or misconfigured alerting mechanisms can result in a failure to promptly notify security teams about potential security incidents. This delay can hinder timely response and mitigation efforts.
6. Lack of Event Correlation: Failure to correlate events and logs from multiple sources can hinder the ability to identify complex attack patterns or understand the full scope of a security incident. Without proper correlation, an attacker's activities may be overlooked.
7. Neglected Monitoring and Review: Inconsistent or infrequent monitoring and review of security logs can lead to missed security events or prolonged exposure to threats. It is important to regularly review logs and assess them against known attack patterns and indicators of compromise.
To address security logging and monitoring failures, organizations should establish robust logging practices, including defining comprehensive logging policies, ensuring proper configuration of log collection mechanisms, and implementing secure log storage and retention practices. Employing automated log analysis tools, implementing real-time alerting mechanisms, and conducting regular log reviews can enhance the effectiveness of security monitoring. Additionally, organizations should regularly assess and update their logging and monitoring strategies to align with evolving security threats and compliance requirements.



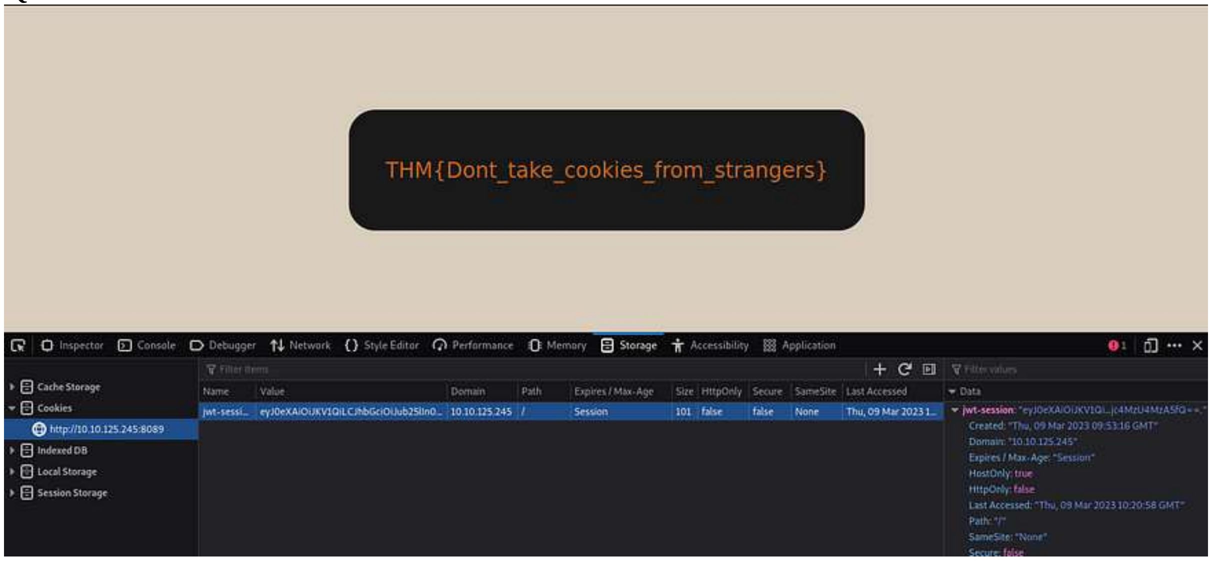
x in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!



Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

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

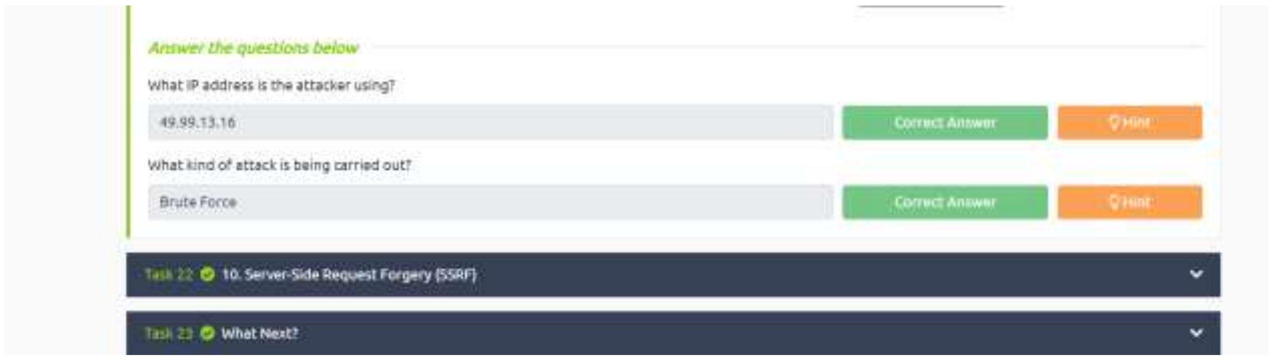
Q4.



THM{Dont_take_cookies_from_strangers}

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
jwt-sessi...	eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0...	10.10.125.245	/	Session	101	false	false	None	Thu, 09 Mar 2023 1...

```
Data
jwt-session: "eyJ0eXAiOiJKV1Qi...jc4MzU4MzA5fQ==."
  Created: "Thu, 09 Mar 2023 09:53:16 GMT"
  Domain: "10.10.125.245"
  Expires / Max-Age: "Session"
  HostOnly: true
  HttpOnly: false
  Last Accessed: "Thu, 09 Mar 2023 10:20:58 GMT"
  Path: "/"
  SameSite: "None"
  Secure: false
```


Answer the questions below

What IP address is the attacker using?

49.99.13.16

What kind of attack is being carried out?

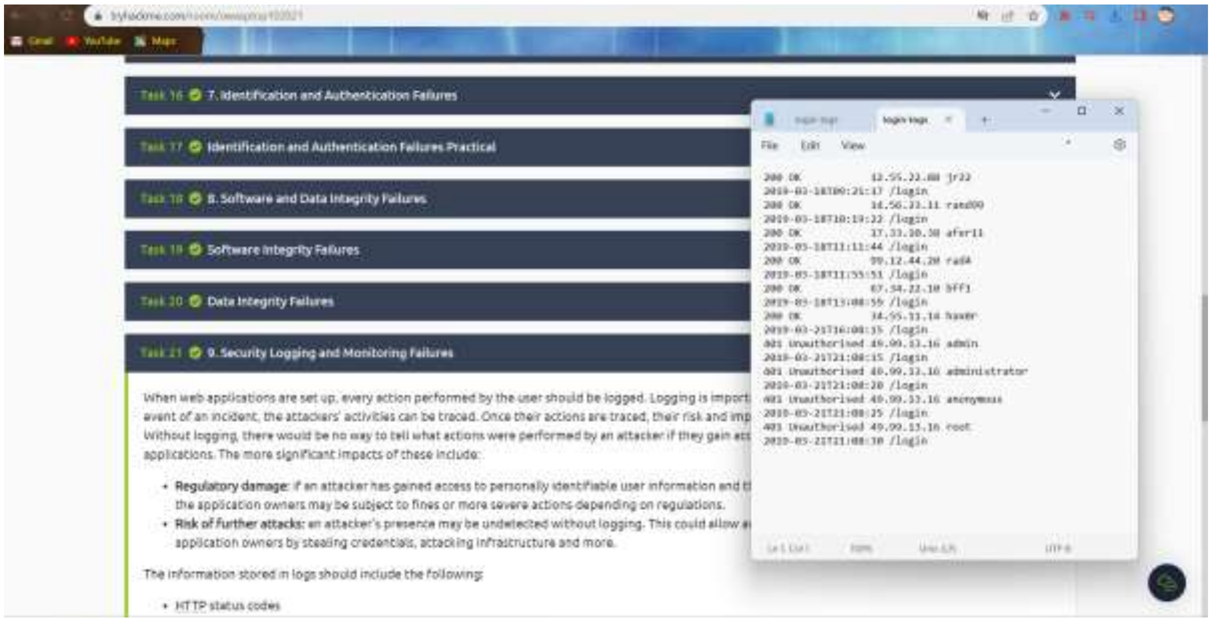
Brute Force

Task 22: Sever-Side Request Forgery

Server-Side Request Forgery (SSRF) is a type of web application vulnerability that allows an attacker to manipulate the server-side functionality of a web application to make unauthorized requests to other internal or external systems. With SSRF, an attacker can abuse the trust placed in the server to access resources or services that should be restricted.
Here's an overview of how SSRF works and its potential impact:
1. Exploiting Input Validation: SSRF typically occurs when a web application fails to properly validate user-supplied input, such as URLs or IP addresses. Attackers can manipulate these inputs to trick the server into making unintended requests.
2. Forging Requests: Once an SSRF vulnerability is identified, an attacker can forge requests to internal resources or external systems. This can include accessing local files, exploiting internal APIs, scanning internal networks, or making requests to arbitrary external hosts.
3. Impact and Consequences: SSRF can have various negative impacts, including unauthorized access to sensitive data, disclosure of internal resources, abuse of internal services, exploitation of vulnerable components, and potential lateral movement within a network.
4. Examples of Exploitation: Attackers can exploit SSRF in different ways. For example, they may redirect the server to access an internal administrative panel, retrieve sensitive files, perform port scanning on internal IP ranges, or initiate attacks on other systems reachable by the server.

To mitigate SSRF vulnerabilities, web application developers and administrators can implement the following best practices:
1. Input Validation: Ensure that all user-supplied inputs are properly validated and sanitized. Use whitelisting techniques to allow only trusted URLs and IP addresses.
2. Restrict Access: Apply strict firewall rules and access controls to limit the resources and services accessible from the server. Avoid allowing arbitrary URLs or IP addresses to be requested from the server.
3. Use Safe APIs: If the application requires making requests to external systems, use safe and specific APIs instead of allowing unrestricted access to arbitrary URLs.
4. Network Segmentation: Implement proper network segmentation to separate critical internal resources from publicly accessible systems. This helps limit the impact of any potential SSRF attacks.
5. Least Privilege: Ensure that the server or application has the least privileged access necessary to perform its intended tasks. Restrict access to sensitive resources and services.
6. Security Testing: Regularly conduct security testing, including vulnerability scanning and

penetration testing, to identify and remediate SSRF vulnerabilities.
By adopting these measures, organizations can significantly reduce the risk of SSRF vulnerabilities and protect their applications and underlying infrastructure from potential exploitation.



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

Task 20 Data Integrity Failures

Task 21 9. Security Logging and Monitoring Failures

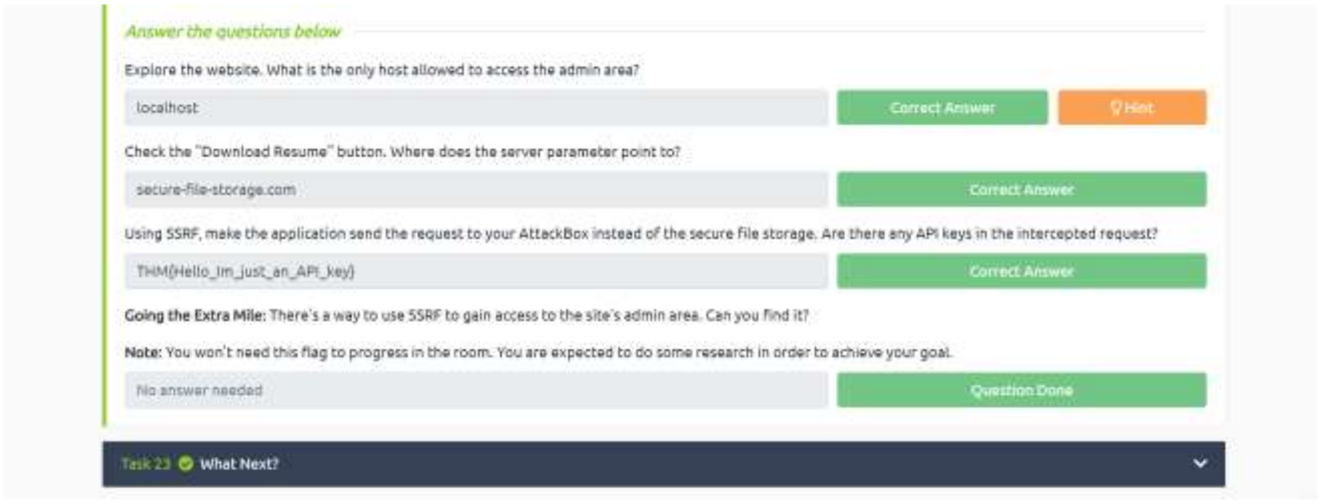
When web applications are set up, every action performed by the user should be logged. Logging is import
event of an incident, the attackers' activities can be traced. Once their actions are traced, their risk and imp
Without logging, there would be no way to tell what actions were performed by an attacker if they gain acc
applications. The more significant impacts of these include:

- **Regulatory damage:** if an attacker has gained access to personally identifiable user information and t
  the application owners may be subject to fines or more severe actions depending on regulations.
- **Risk of further attacks:** an attacker's presence may be undetected without logging. This could allow a
  application owners by stealing credentials, attacking infrastructure and more.

The information stored in logs should include the following:

- HTTP status codes

```
200 OK           12.55.22.88 jr22
2019-03-18T09:21:17 /login
200 OK           14.56.23.11 rand99
2019-03-18T10:19:22 /login
200 OK           17.33.10.38 afer11
2019-03-18T11:11:44 /login
200 OK           99.12.44.20 rad4
2019-03-18T11:55:51 /login
200 OK           67.34.22.10 bff1
2019-03-18T13:08:59 /login
200 OK           34.55.11.14 hax0r
2019-03-21T16:08:15 /login
401 Unauthorised 49.99.13.16 admin
2019-03-21T21:08:15 /login
401 Unauthorised 49.99.13.16 administrator
2019-03-21T21:08:20 /login
401 Unauthorised 49.99.13.16 anonymous
2019-03-21T21:08:25 /login
401 Unauthorised 49.99.13.16 root
2019-03-21T21:08:30 /login
```



Answer the questions below

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

localhost

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

secure-file-storage.com

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}

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