OWASP Top 10 - 2021

Introduction

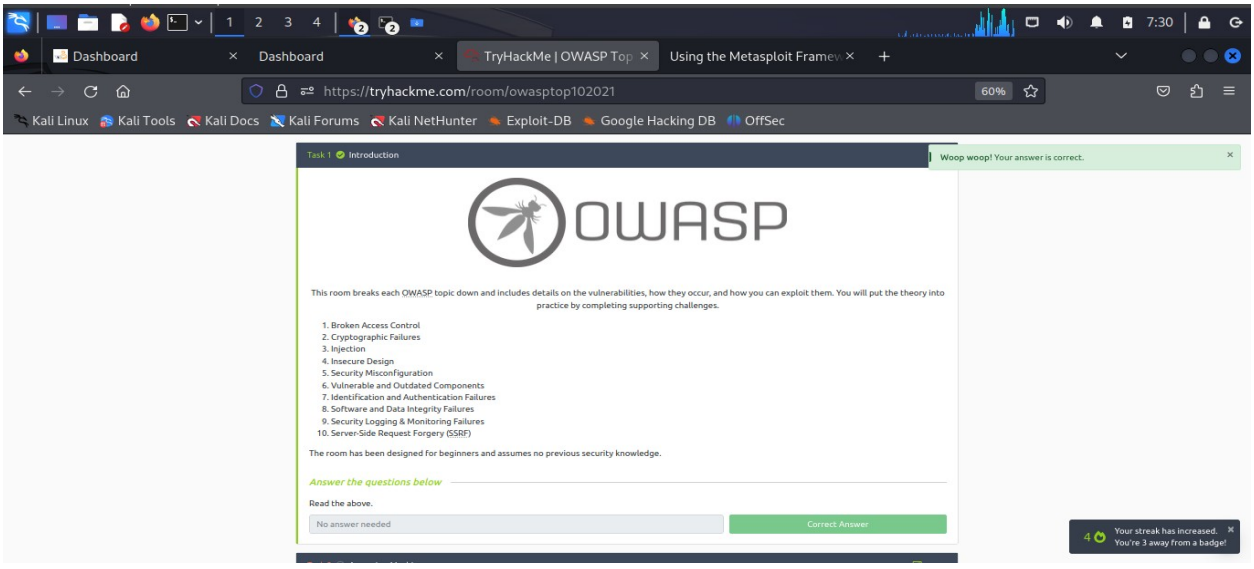
This sub task introduces the learner to the Open Web Application Security Project Top 10 topic down and includes details on the vulnerabilities, how they occur, and how you can exploit them. (OWASP) of Cyber Security by going through a series of what is called rooms for a special skill particularly covering the following areas:

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 Vulnerable and Outdated Components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. Software and Data Integrity Failures
9. Security Logging & Monitoring Failures
10. Server-Side Request Forgery (SSRF)

Activities

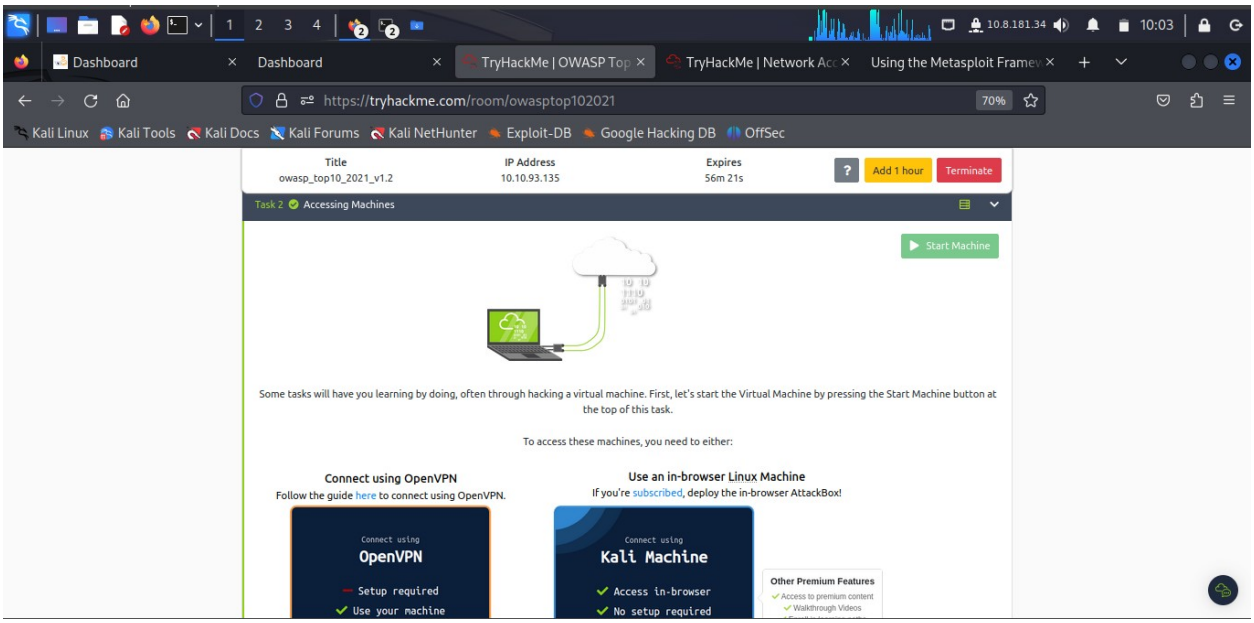
Task 1: Introduction

This section outlines objectives such as Broken Access Control using a virtual machine, cryptographic Failures, injection insecure design, security misconfiguration and much more.



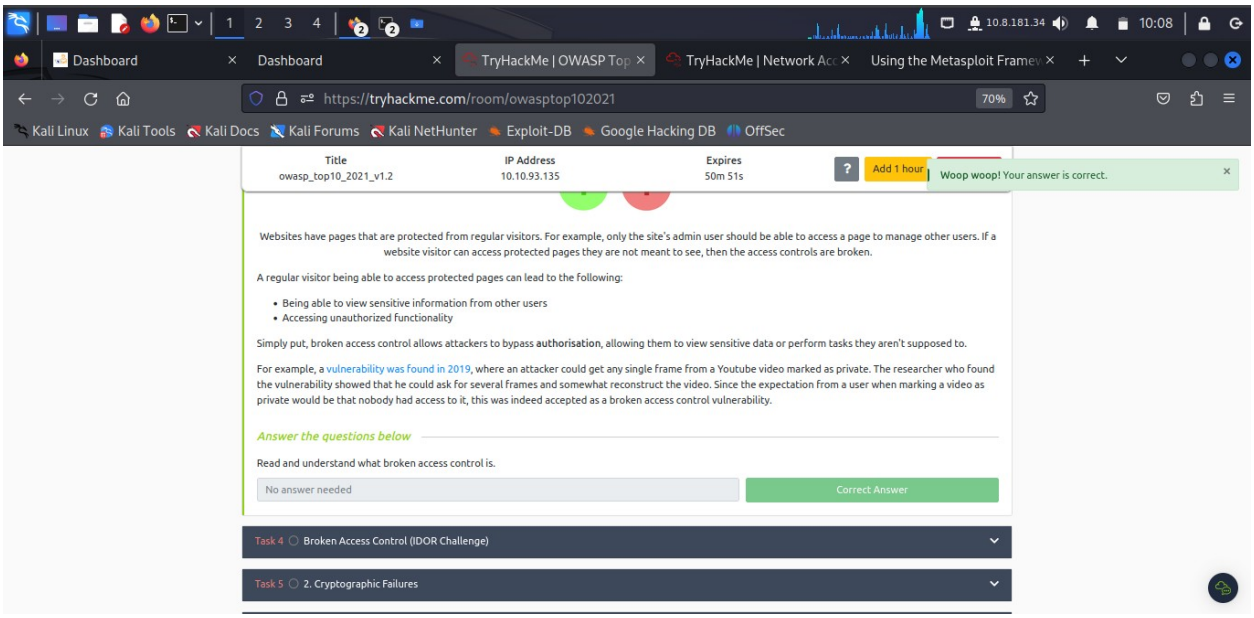
Task 2: Accessing Machines

This is where the learner activates the VM to conduct practical cases taught.



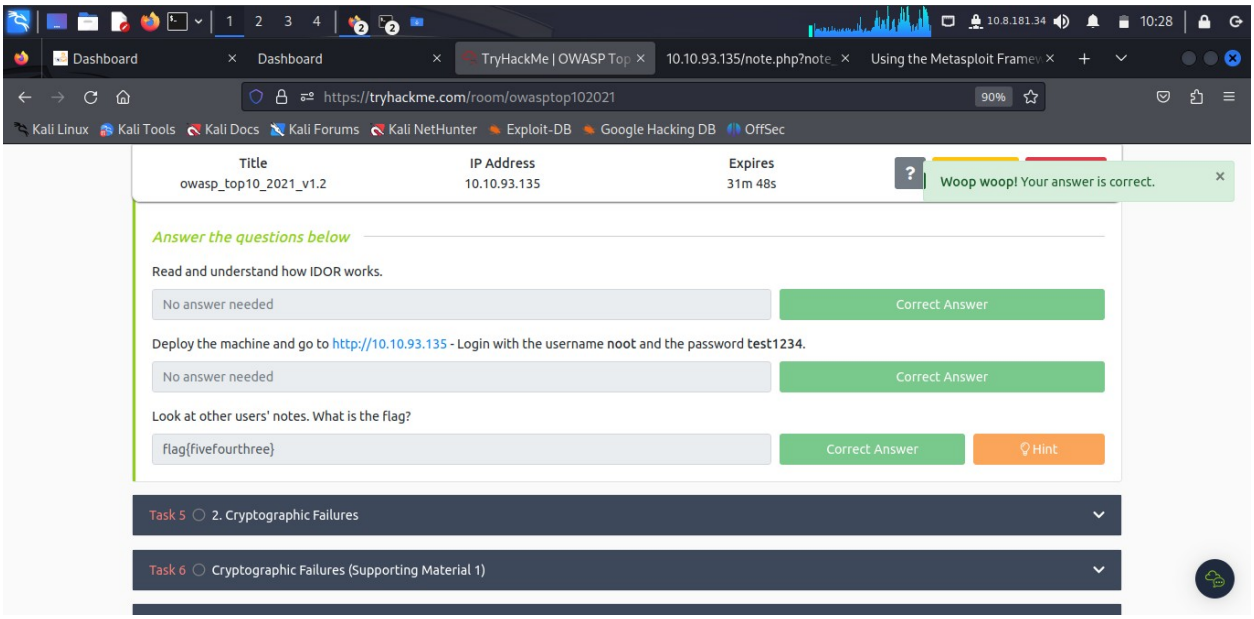
Task 3: 1. Broken Access Control

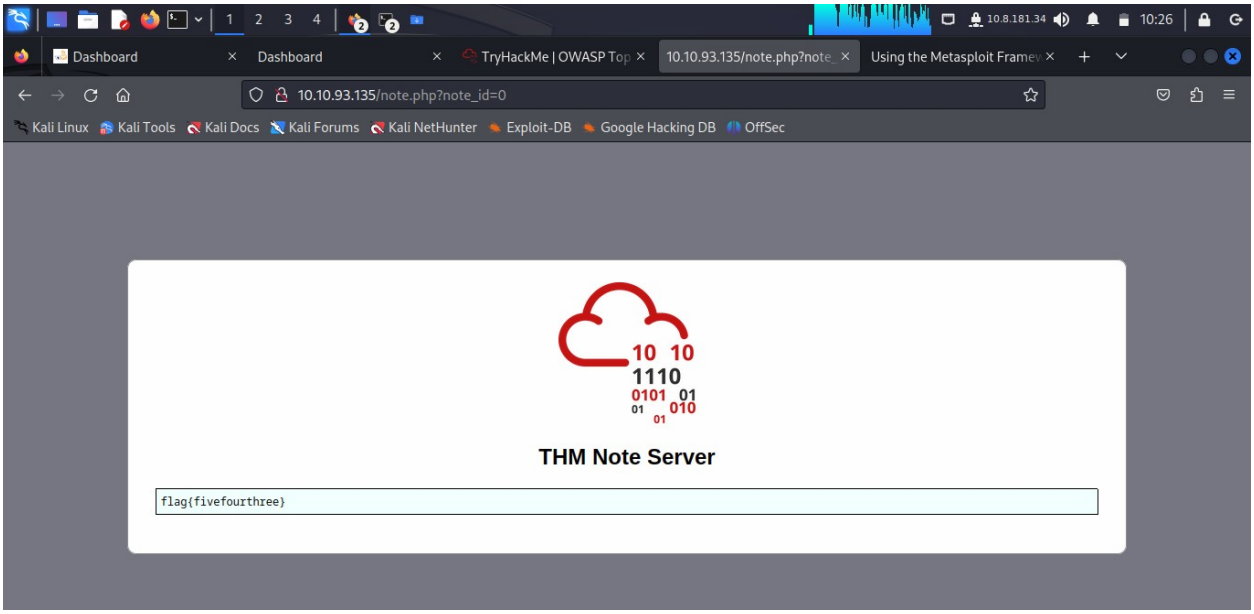
The learner goes deep to know what broken access control can lead to unauthorized privilages that is, allowing attackers to bypass **authorisation**, allowing them to view sensitive data or perform tasks they aren't supposed to.



Task 4: Broken Access Control (IDOR Challenge)

Insecure Direct Object Reference (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***, - an identifier referring to specific objects (a file, a user, a bank account in a banking application) within the server.



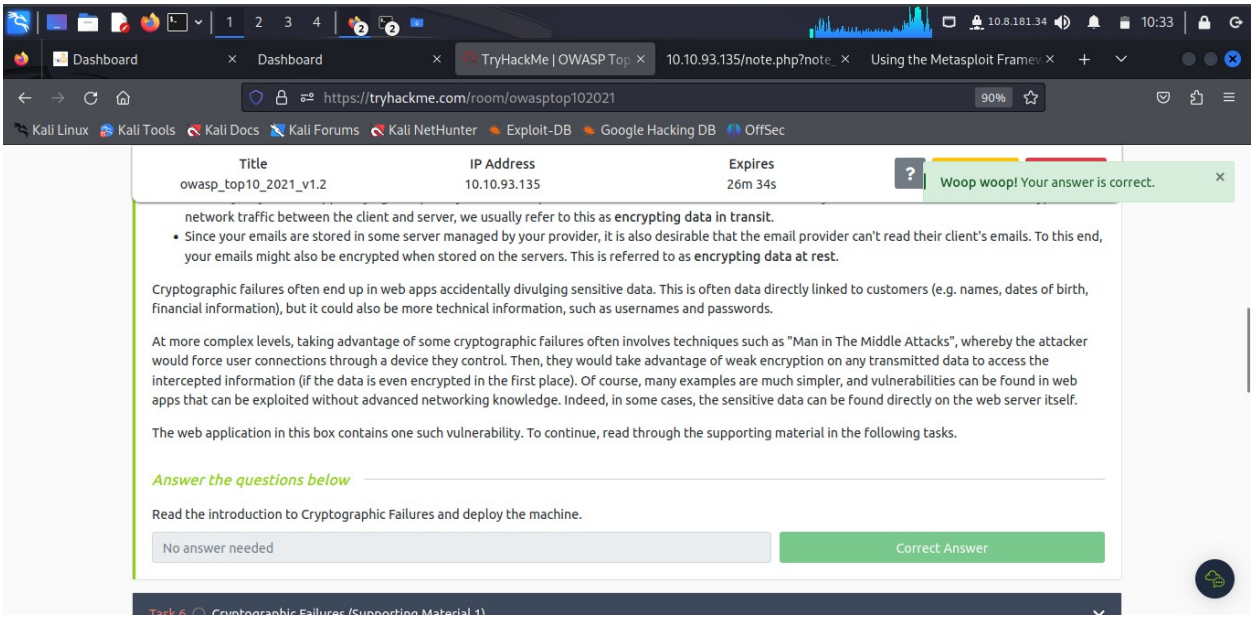


The learner accessed the above information by visiting http://10.10.93.135 - with the username as **noot** and the password as **test1234**.

Task 5: 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.

The learner now understand is entailed under cryptographic failures in which they often end up in web apps accidentally divulging sensitive data. This is often data directly linked to customers (e.g. names, dates of birth, financial information), but it could also be more technical information, such as usernames and passwords.



Task 6: Cryptographic Failures (Supporting Material 1)

Large amount of data is stored in databases that necessitates wider accessibility for multiple user locations.

Web application, as many users may interact with the website at any time usually follow the Structured Query Language (SQL) syntax.

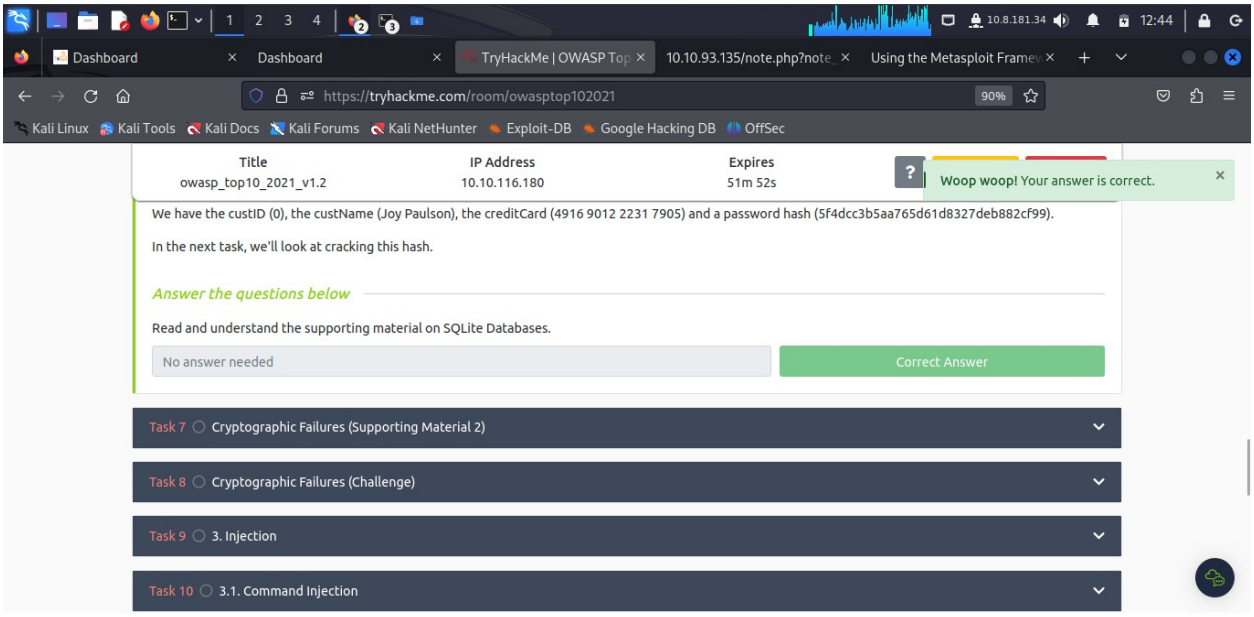
Databases set up on dedicated servers running a database service such as MySQL or MariaDB; however, databases can also be stored as files. These are referred to as **flat-file** databases, as they are stored as a single file on the computer. This is much easier than setting up an entire database server and could potentially be seen in smaller web applications.

A good case is a web application through interaction with SQL syntax.

Flat-file databases are stored as a file on the disk of a computer. If the database is stored underneath the root directory of the website. The file can be downloaded and queried on a local machine, with full access to everything in the database.

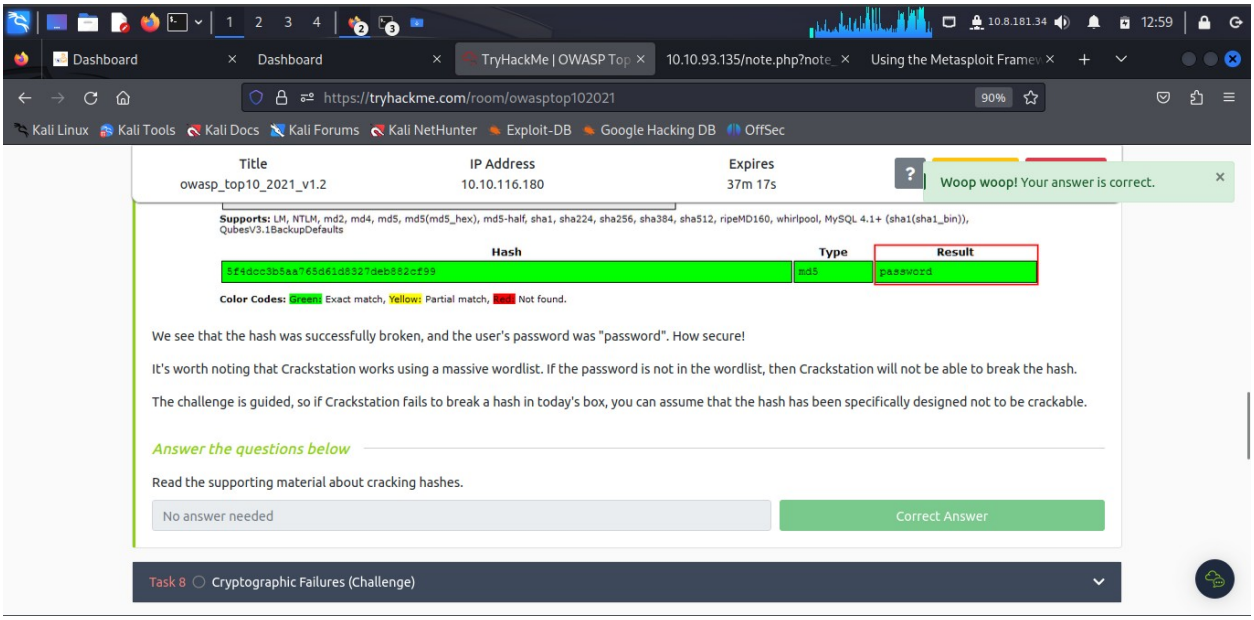
SQLite database is the most common (and simplest) format of a flat-file.

The client is **sqlite3** and installed on many Linux distributions.



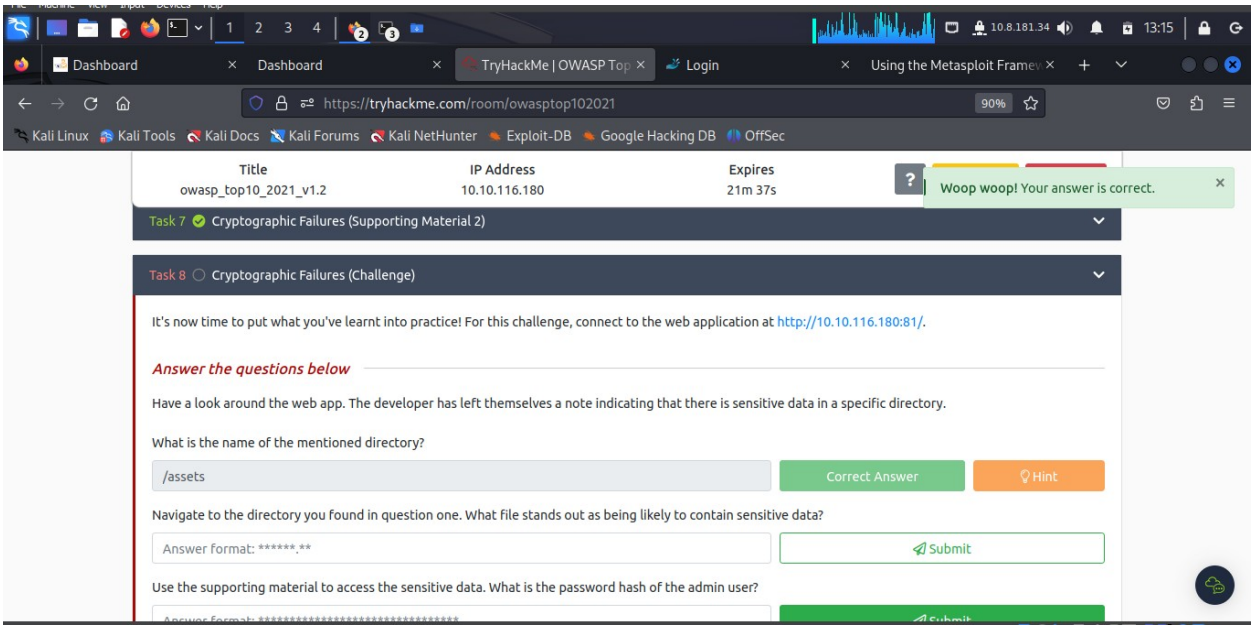
Task 7: Cryptographic Failures (Supporting Material 2)

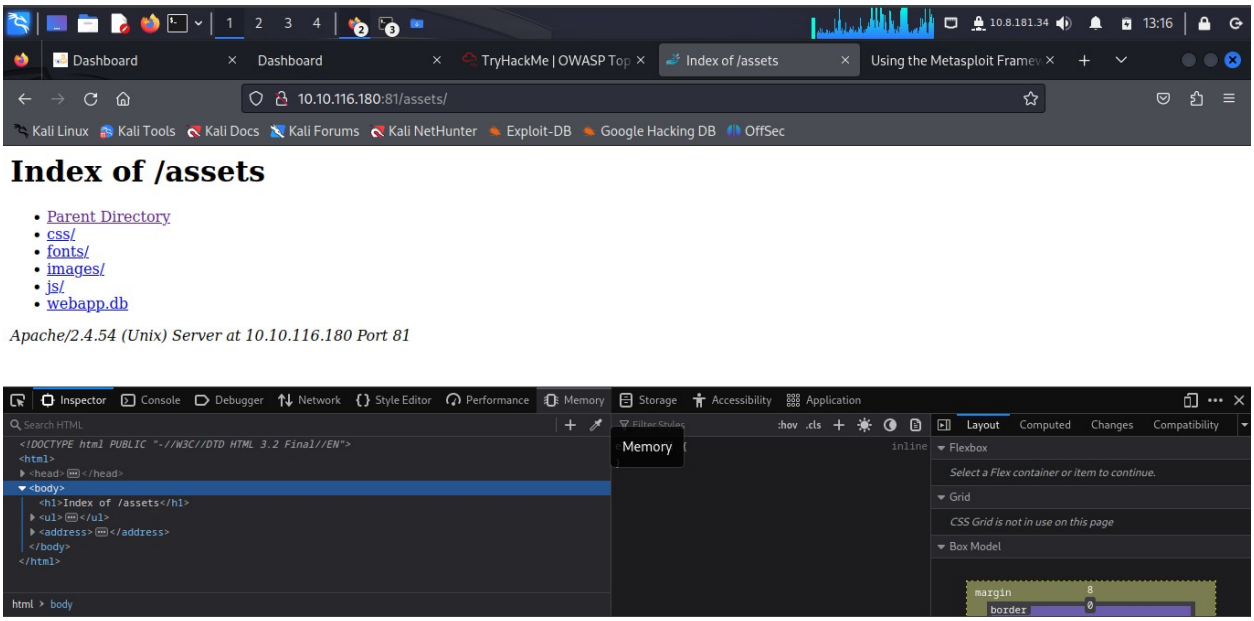
The learner understands that Kali comes pre-installed with various tools for hash cracking. Also there is online platform - the online tool: Crackstation which enables cracking weak password hashes.



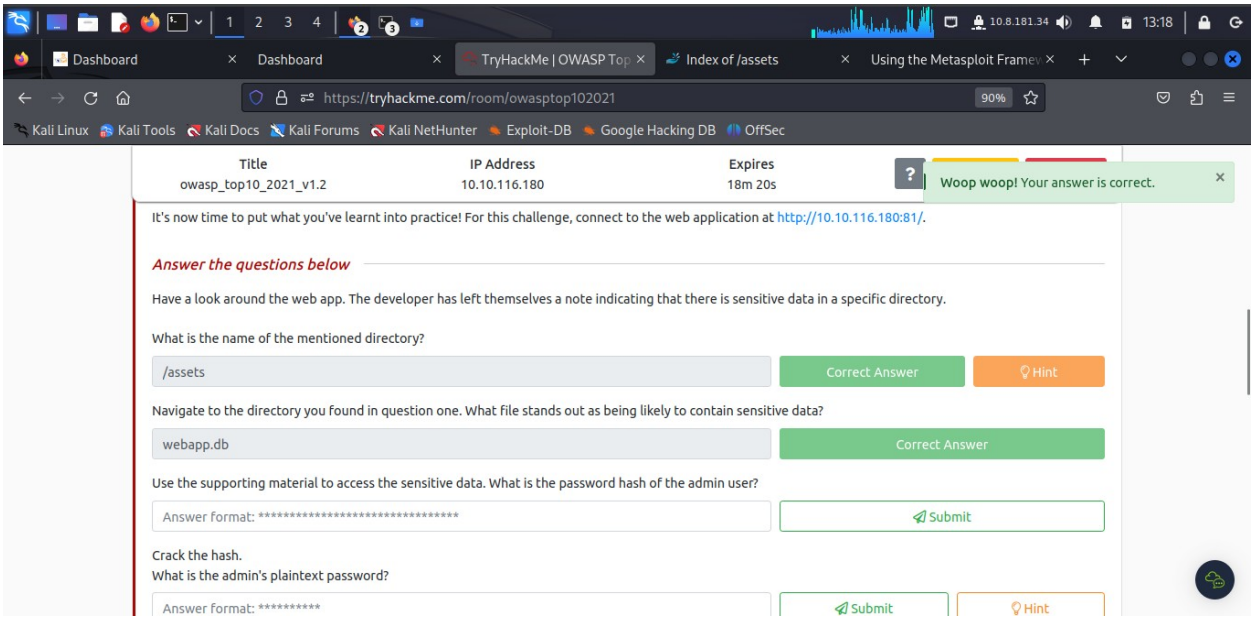
Task 8: Cryptographic Failures (Challenge)

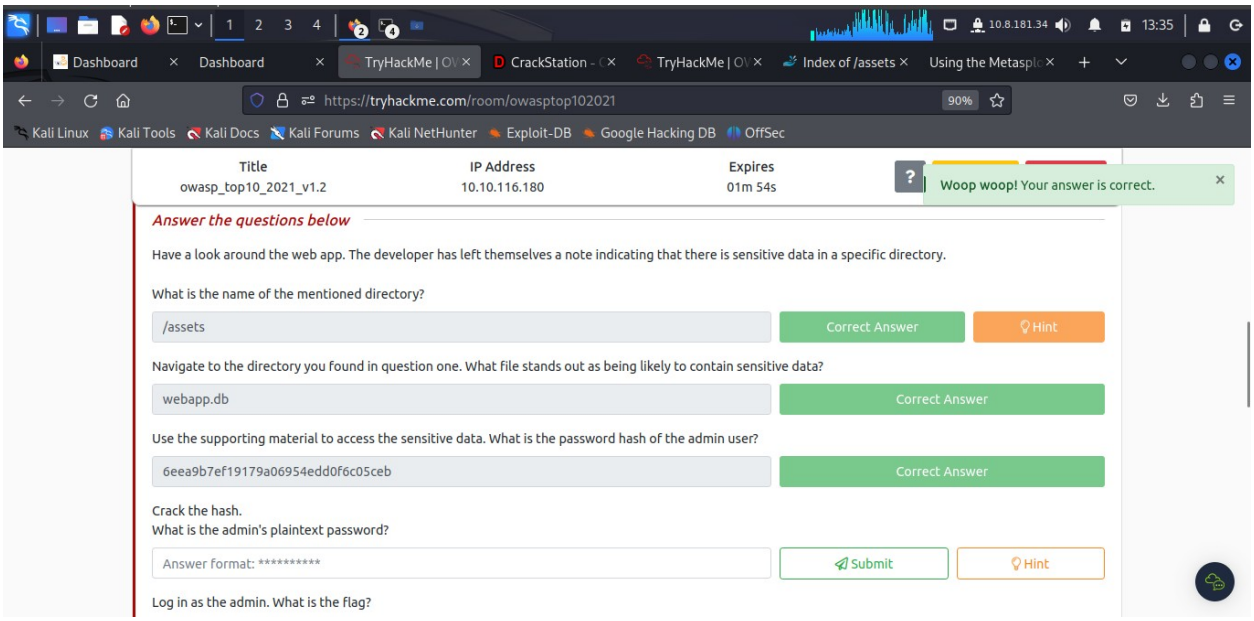
The learner accessed **/assets** after inpecting the login page.



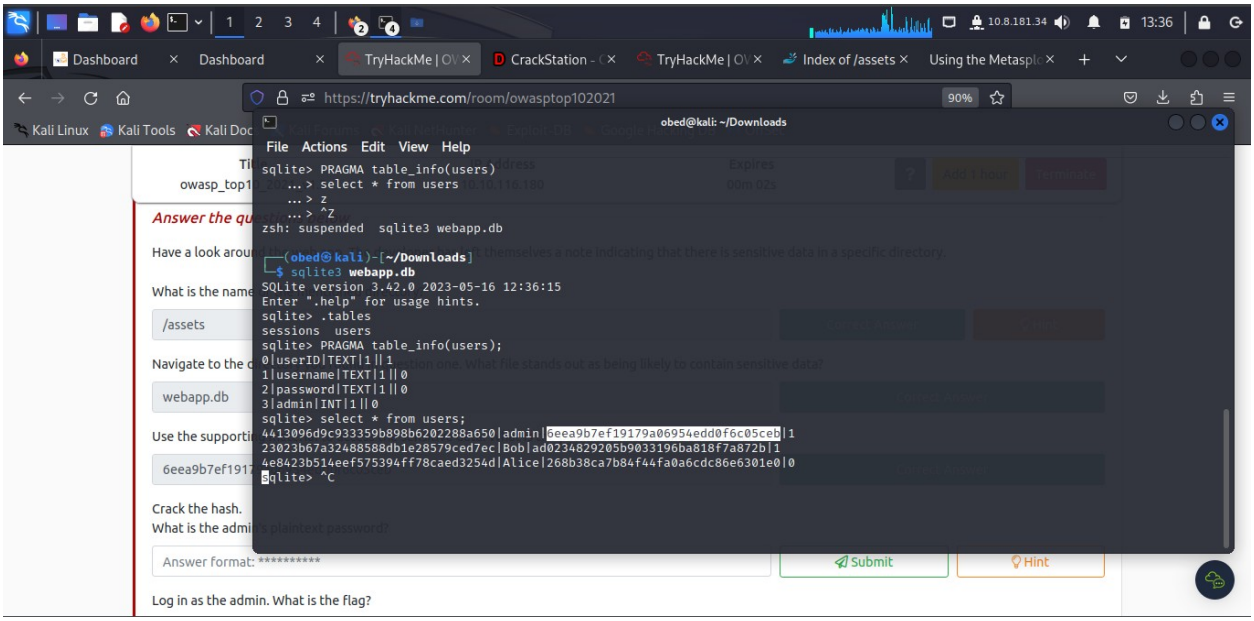


Navigating to **/assets** will give us the **webapp.db** files among others.

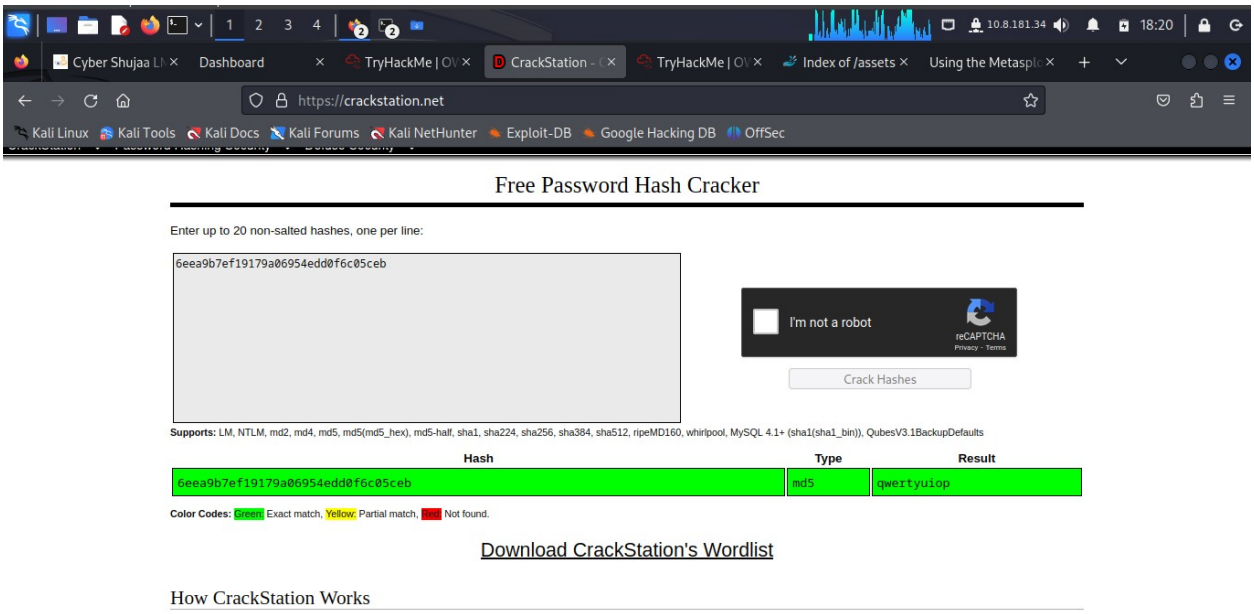


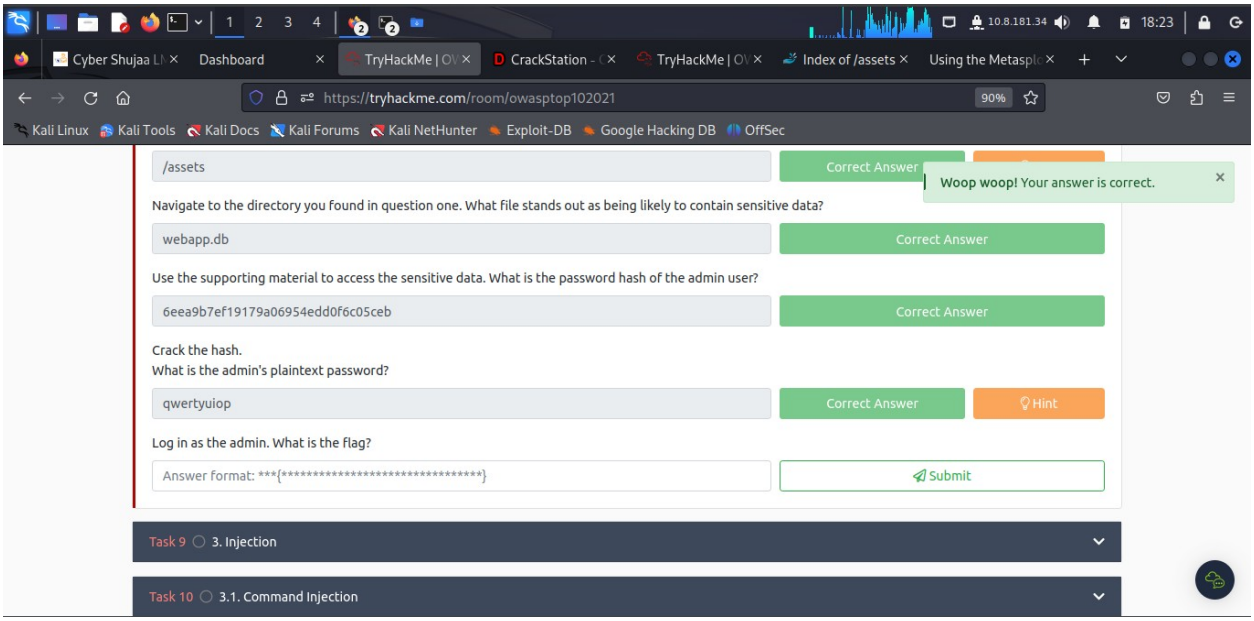


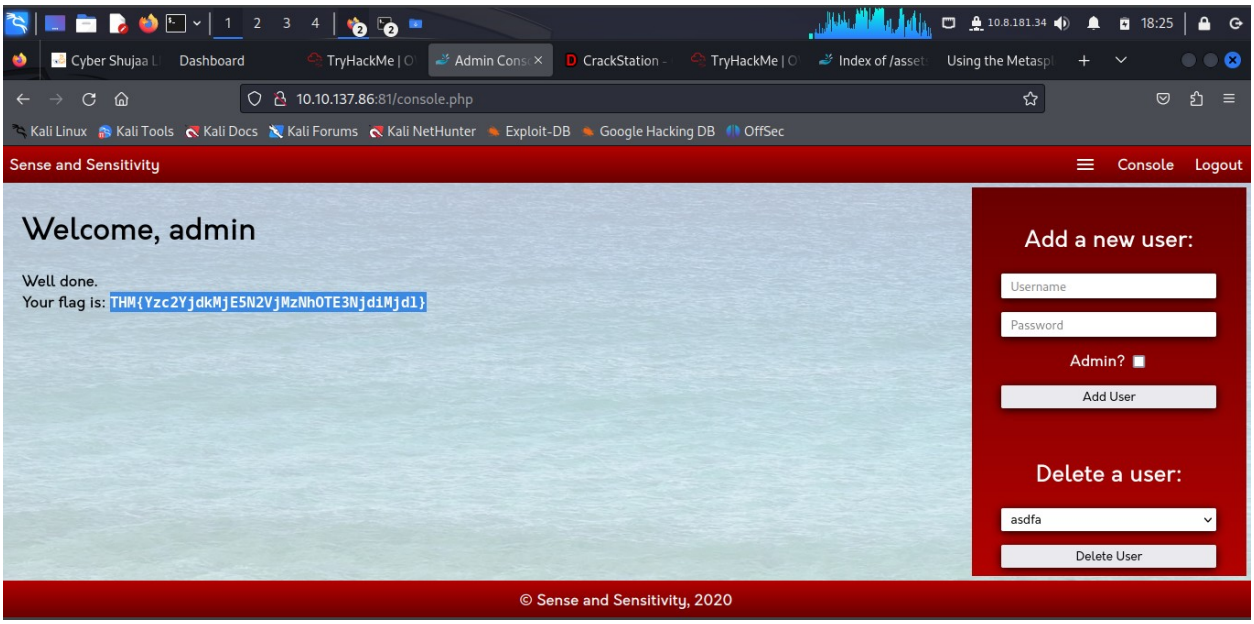
The learner run the **sqlite3** and accessed the file **webapp.db** located in ***Download*** directory.

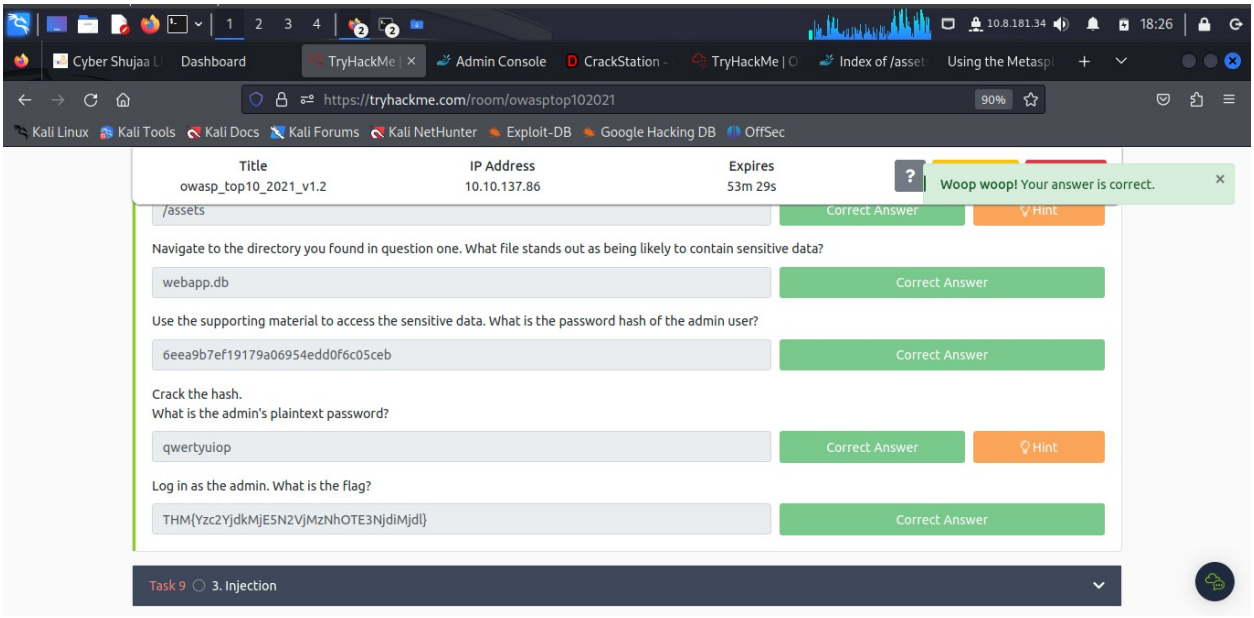


To crack the hash code, the learner used ***CrackStation***




https://tryhackme.com/room/owasptop102021
/assets
Correct Answer
Woop woop! Your answer is correct.
Navigate to the directory you found in question one. What file stands out as being likely to contain sensitive data?
webapp.db
Correct Answer
Use the supporting material to access the sensitive data. What is the password hash of the admin user?
6eea9b7ef19179a06954edd0f6c05ceb
Correct Answer
Crack the hash.
What is the admin's plaintext password?
qwertyuiop
Correct Answer
Hint
Log in as the admin. What is the flag?
Answer format: ***{**********}
Submit
Task 9 3. Injection
Task 10 3.1. Command Injection


10.10.137.86:81/console.php
Sense and Sensitivity
Console
Logout
Welcome, admin
Well done.
Your flag is: THM{Yzc2YjdkMjE5N2VjMzNhOTE3NjdiMjdl}
Add a new user:
Username
Password
Admin?
Add User
Delete a user:
asdfa
Delete User
© Sense and Sensitivity, 2020



Task 9: 3. Injection

Injection flaws occur because the application interprets user-controlled input as commands or parameters. Injection attacks depend on what technologies are used and how these technologies interpret the input. Some common examples include:

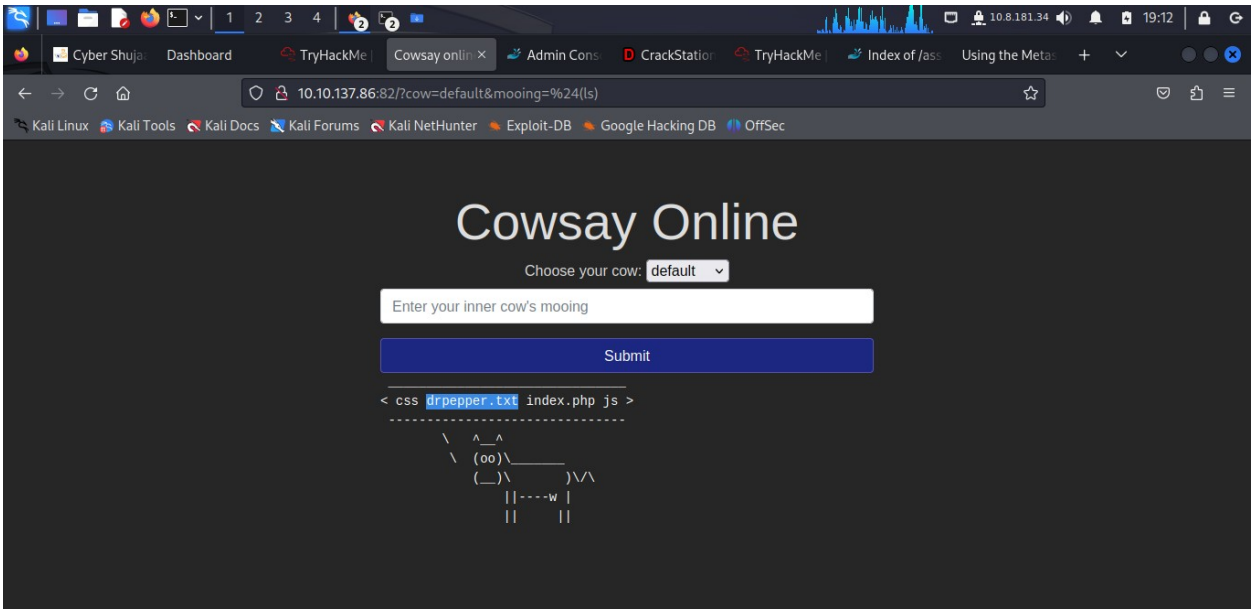
- **SQL Injection:** This occurs when user-controlled input is passed to SQL queries. As a result, an attacker can pass in SQL queries to manipulate the outcome of such queries. This could potentially allow the attacker to access, modify and delete information in a database when this input is passed into database queries. This would mean an attacker could steal sensitive information such as personal details and credentials.
- **Command Injection:** This 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.

To counter this SQL injection is ensuring that user-controlled input is not interpreted as queries or commands. There are different ways of doing this:

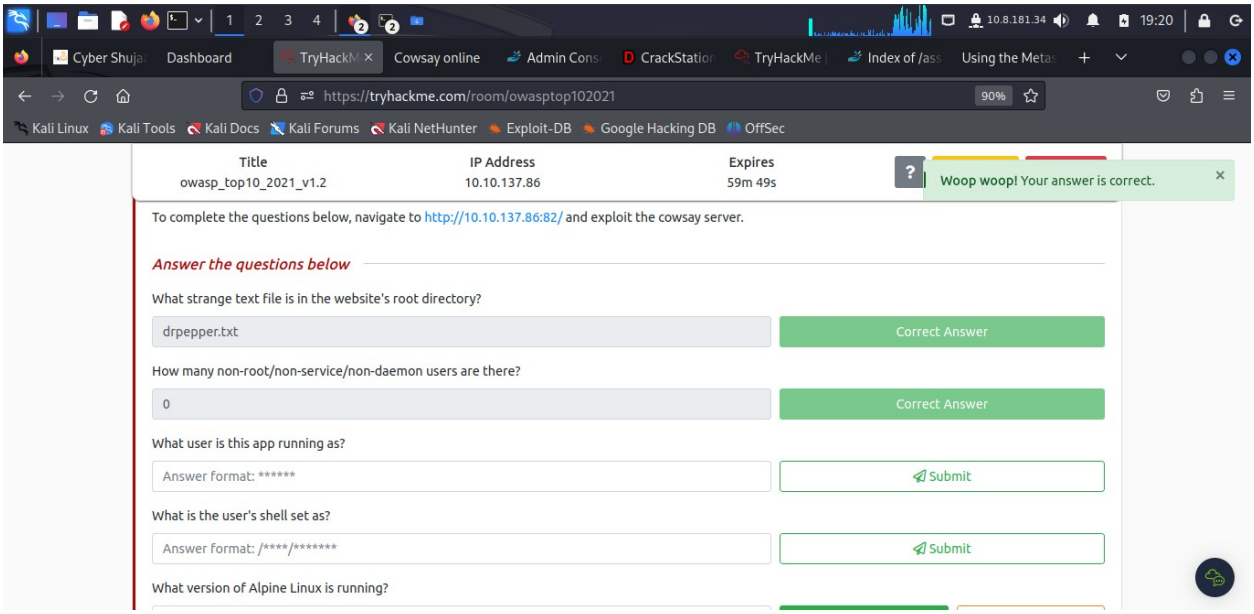
- **Using an allow list:** when input is sent to the server, this input is compared to a list of safe inputs or characters. If the input is marked as safe, then it is processed. Otherwise, it is rejected, and the application throws an error.
- **Stripping input:** If the input contains dangerous characters, these are removed before processing.

Task 10: 3.1. Command Injection

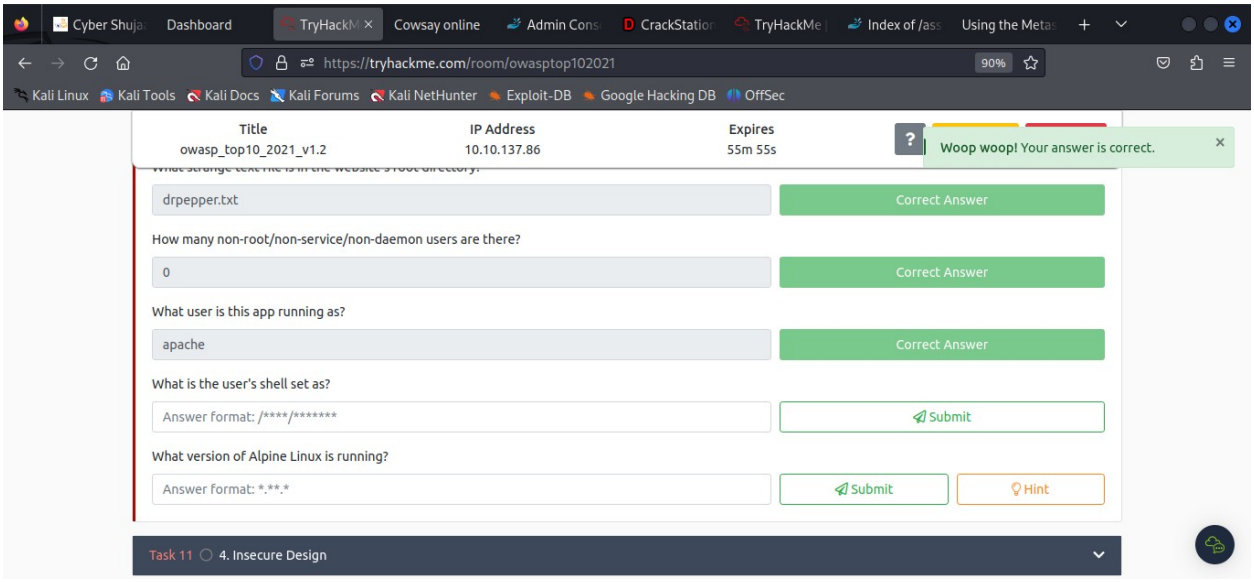
Command Injection occurs when server-side code (like PHP) in a web application makes a call to a function that interacts with the server's console directly. An injection web vulnerability allows an attacker to take advantage of that call to execute operating system commands arbitrarily on the server.

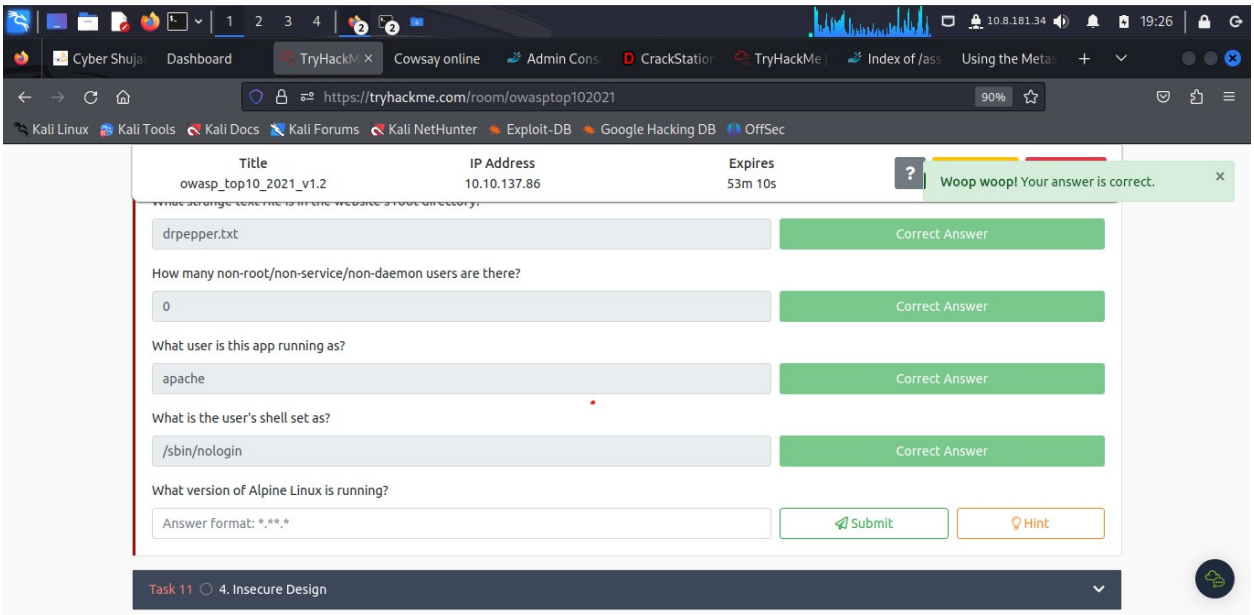


The learner run **$(ls)**

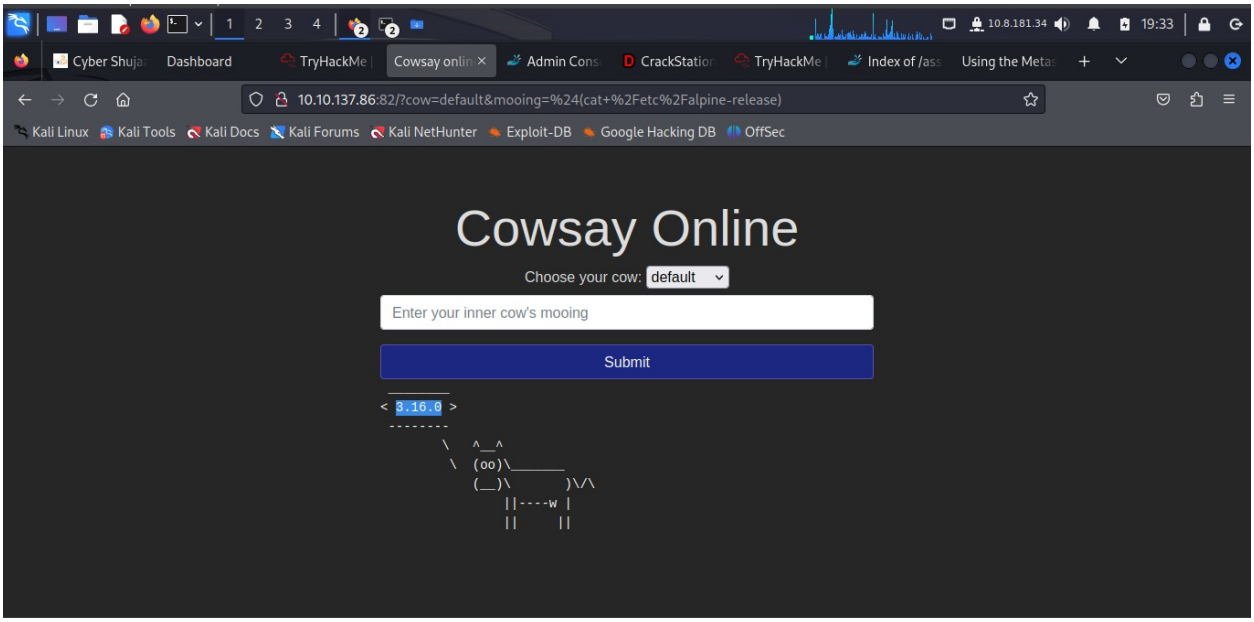


To find out the answer to the question the learner run **$(cat /etc/passwd)**





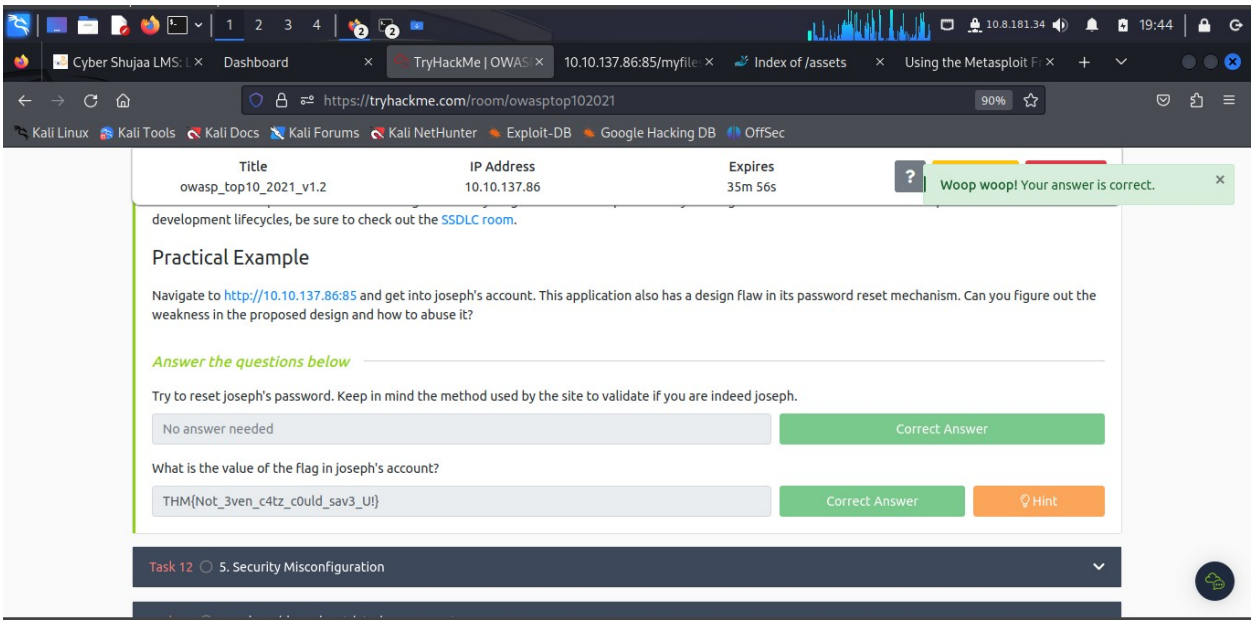
To find the user's shell the learner run **$(cat /etc/passwd)** and the shell is **/sbin/nologin**



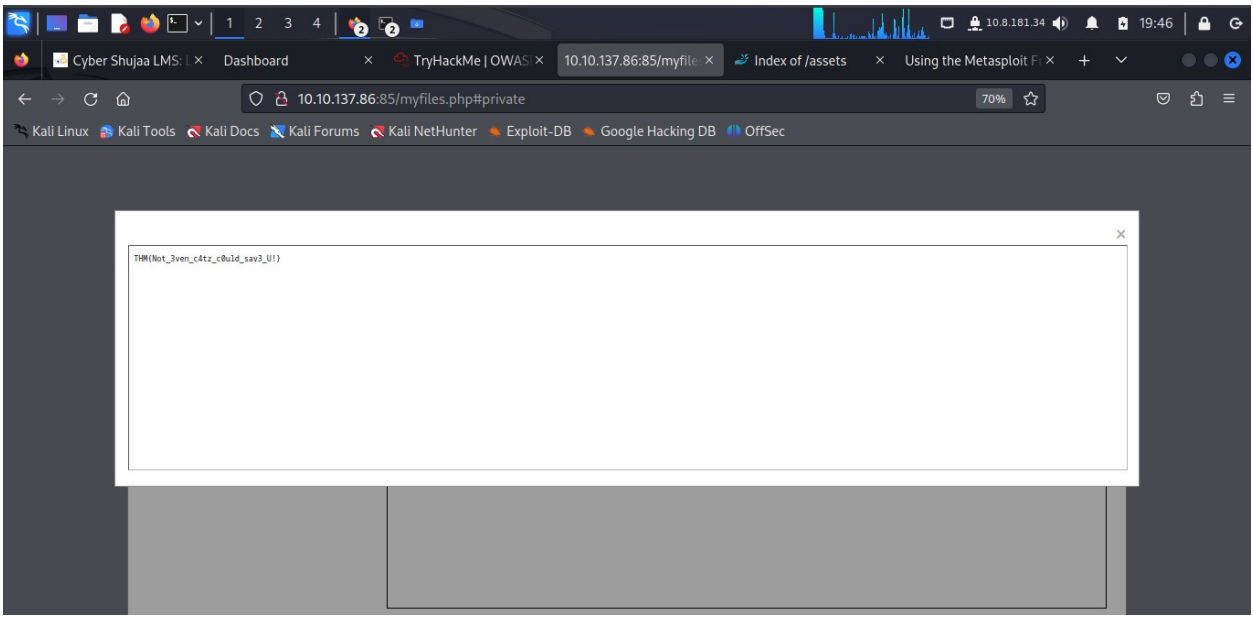
To get the Alpine Linux version the learner run **$(cat /etc/alpine-release)** and the response found is **3.16.0**

Task 11: 4. Insecure Design

This 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.



The learner made a bruteforce attempts severally to get the password and accessed the joseph's account. **THM{Not_3ven_c4tz_c0uld_sav3_U!}**

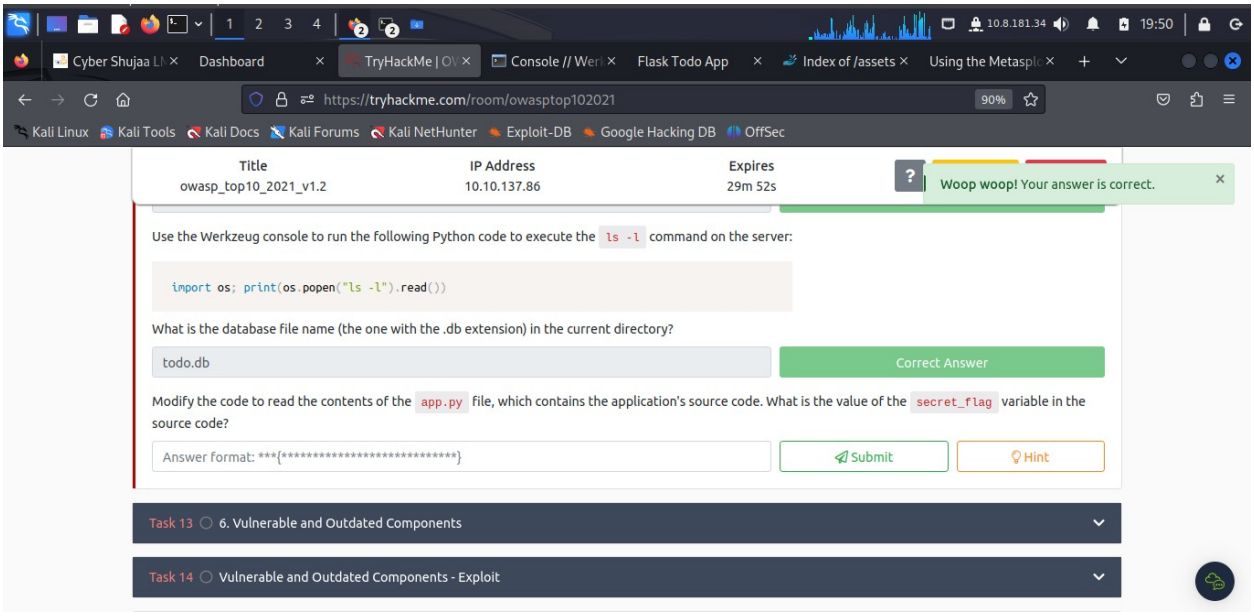


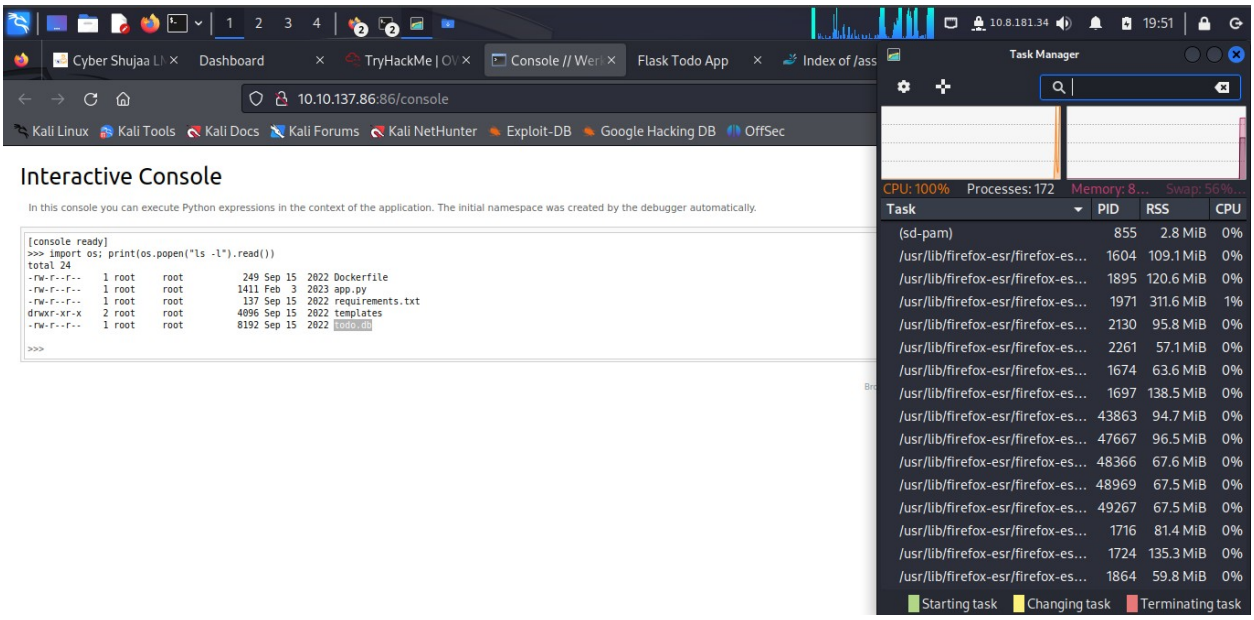
Task 12: 5. Security Misconfiguration

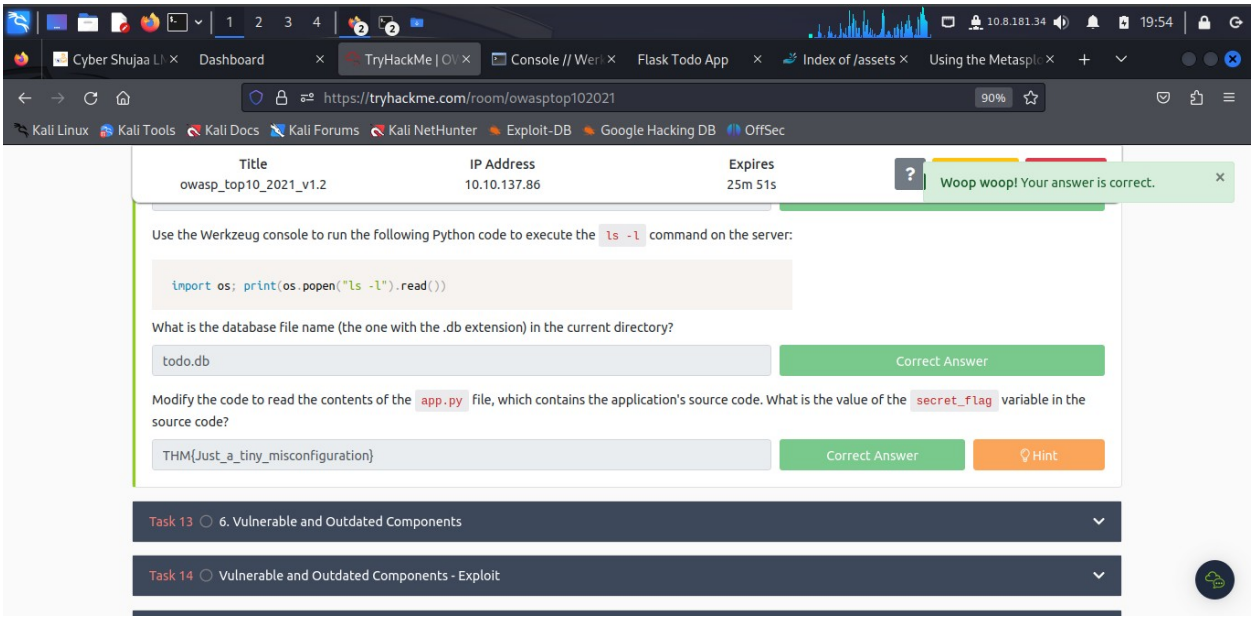
These distinct from the other Top 10 vulnerabilities because they occur when security could have been appropriately configured but was not. Even if downloading the latest up-to-date software, poor configurations could make your installation vulnerable.

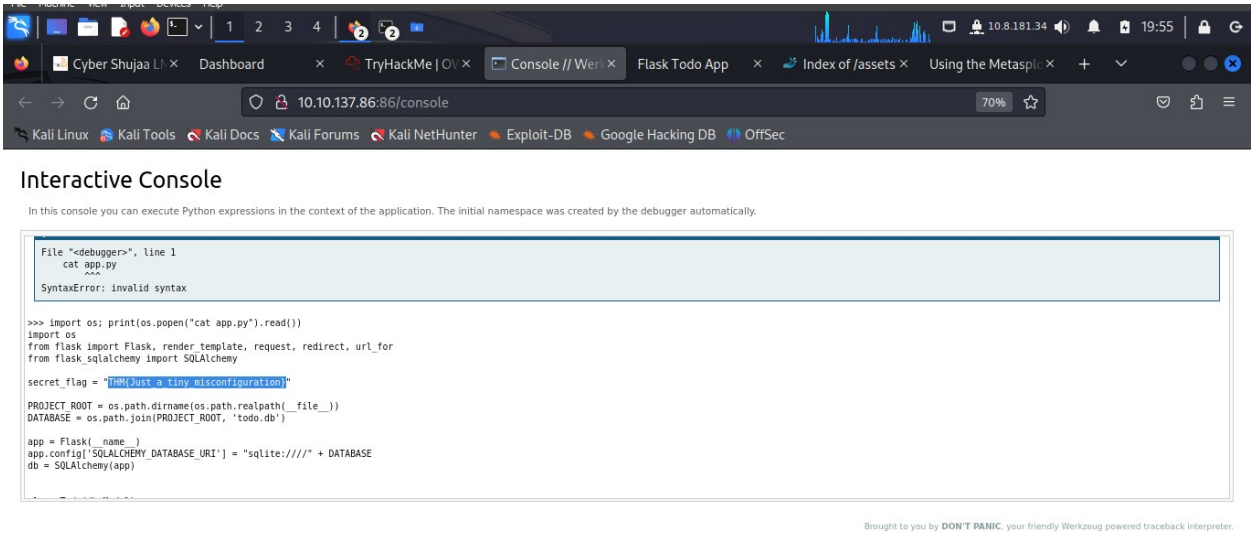
They include:

- 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.

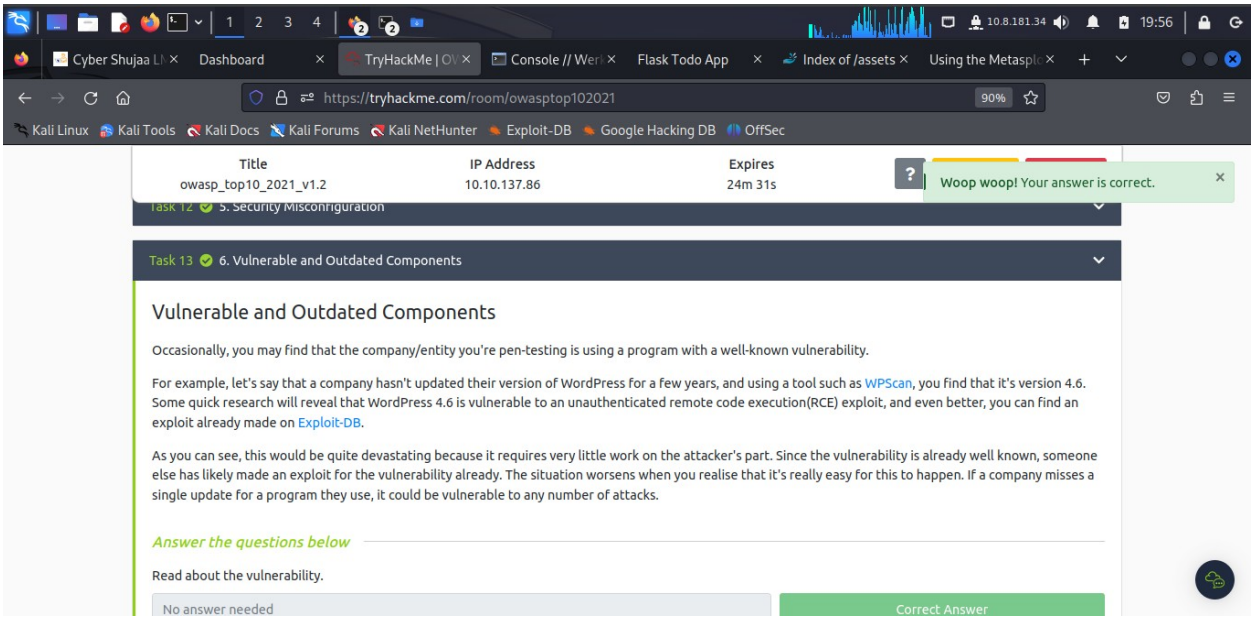

Title
owasp_top10_2021_v1.2
IP Address
10.10.137.86
Expires
29m 52s
Woop woop! Your answer is correct.
Use the Werkzeug console to run the following Python code to execute the ls -l command on the server:
import os; print(os.popen("ls -l").read())
What is the database file name (the one with the .db 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?
Answer format: ***{**********}
Submit
Hint
Task 13 6. Vulnerable and Outdated Components
Task 14 Vulnerable and Outdated Components - Exploit


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 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
>>>
Task Manager
CPU: 100% Processes: 172 Memory: 8... Swap: 56%...
Task PID RSS CPU
(sd-pam) 855 2.8 MiB 0%
/usr/lib/firefox-esr/firefox-es... 1604 109.1 MiB 0%
/usr/lib/firefox-esr/firefox-es... 1895 120.6 MiB 0%
/usr/lib/firefox-esr/firefox-es... 1971 311.6 MiB 1%
/usr/lib/firefox-esr/firefox-es... 2130 95.8 MiB 0%
/usr/lib/firefox-esr/firefox-es... 2261 57.1 MiB 0%
/usr/lib/firefox-esr/firefox-es... 1674 63.6 MiB 0%
/usr/lib/firefox-esr/firefox-es... 1697 138.5 MiB 0%
/usr/lib/firefox-esr/firefox-es... 43863 94.7 MiB 0%
/usr/lib/firefox-esr/firefox-es... 47667 96.5 MiB 0%
/usr/lib/firefox-esr/firefox-es... 48366 67.6 MiB 0%
/usr/lib/firefox-esr/firefox-es... 48969 67.5 MiB 0%
/usr/lib/firefox-esr/firefox-es... 49267 67.5 MiB 0%
/usr/lib/firefox-esr/firefox-es... 1716 81.4 MiB 0%
/usr/lib/firefox-esr/firefox-es... 1724 135.3 MiB 0%
/usr/lib/firefox-esr/firefox-es... 1864 59.8 MiB 0%
Starting task
Changing task
Terminating task



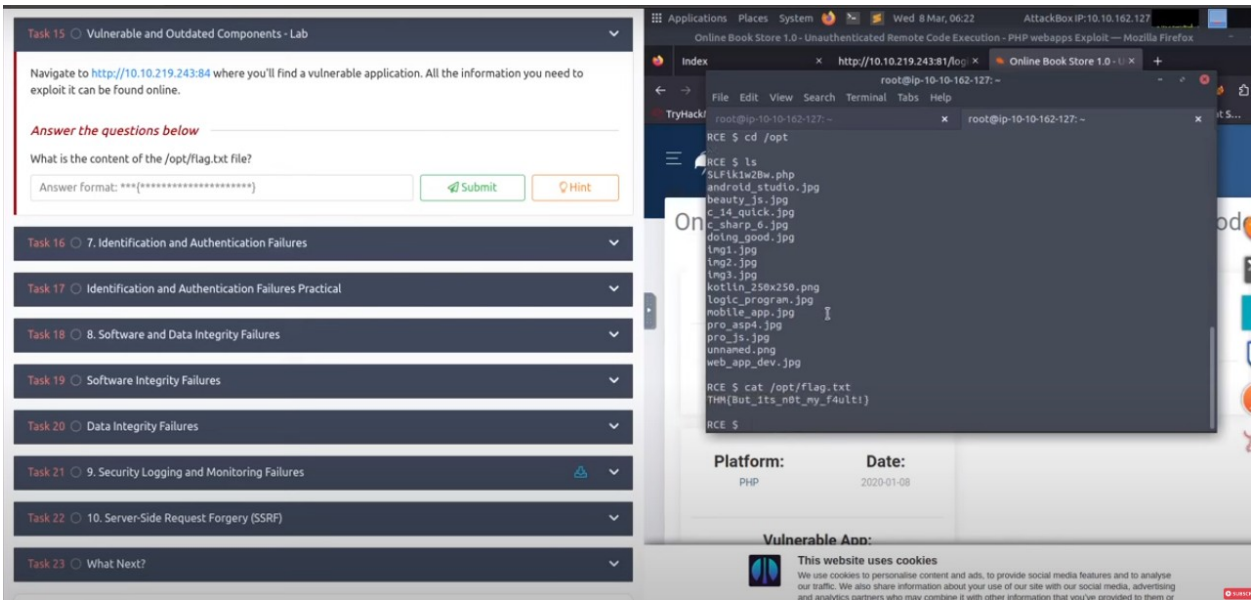


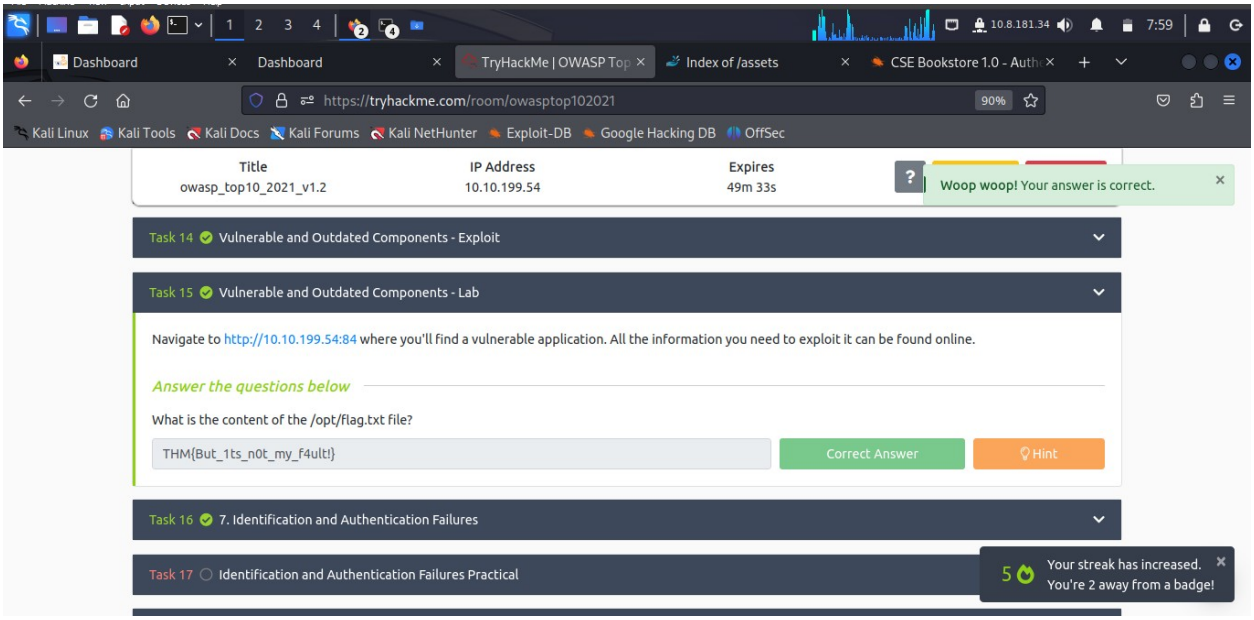
Task 13: 6. Vulnerable and Outdated Components



Task 14: Vulnerable and Outdated Components – Exploit

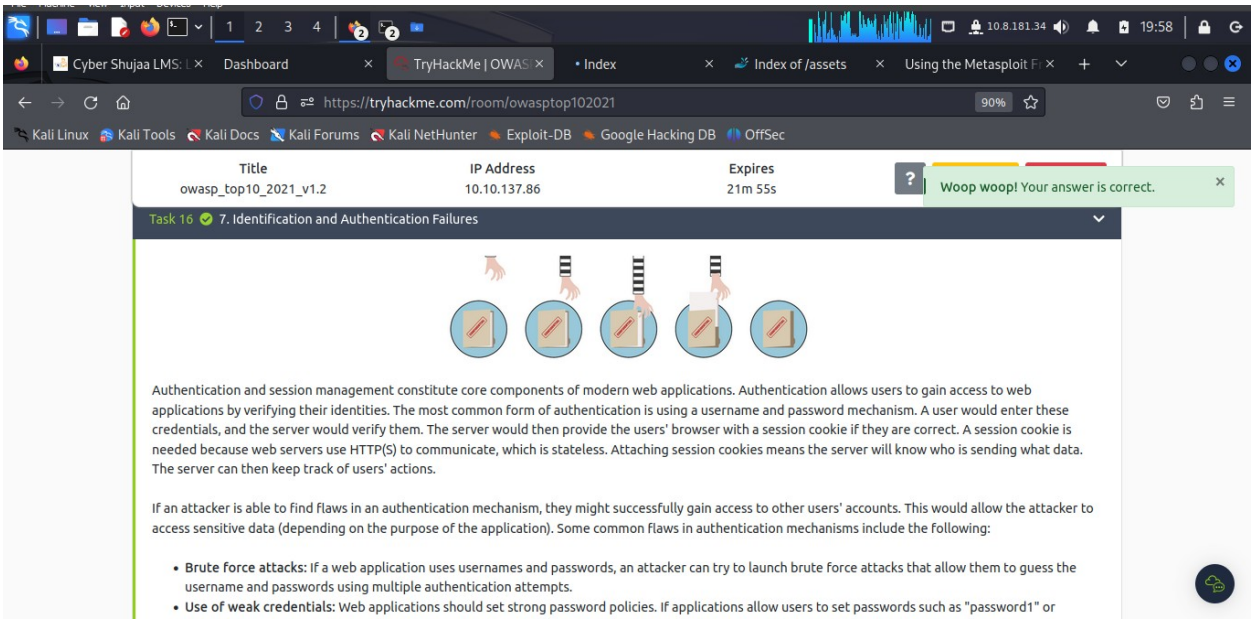
Task 15: Vulnerable and Outdated Components - Lab





Task 16: 7. 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.



If an attacker is able to find flaws in an authentication mechanism, they might successfully gain access to other users' accounts. Some common flaws in authentication mechanisms include the following:

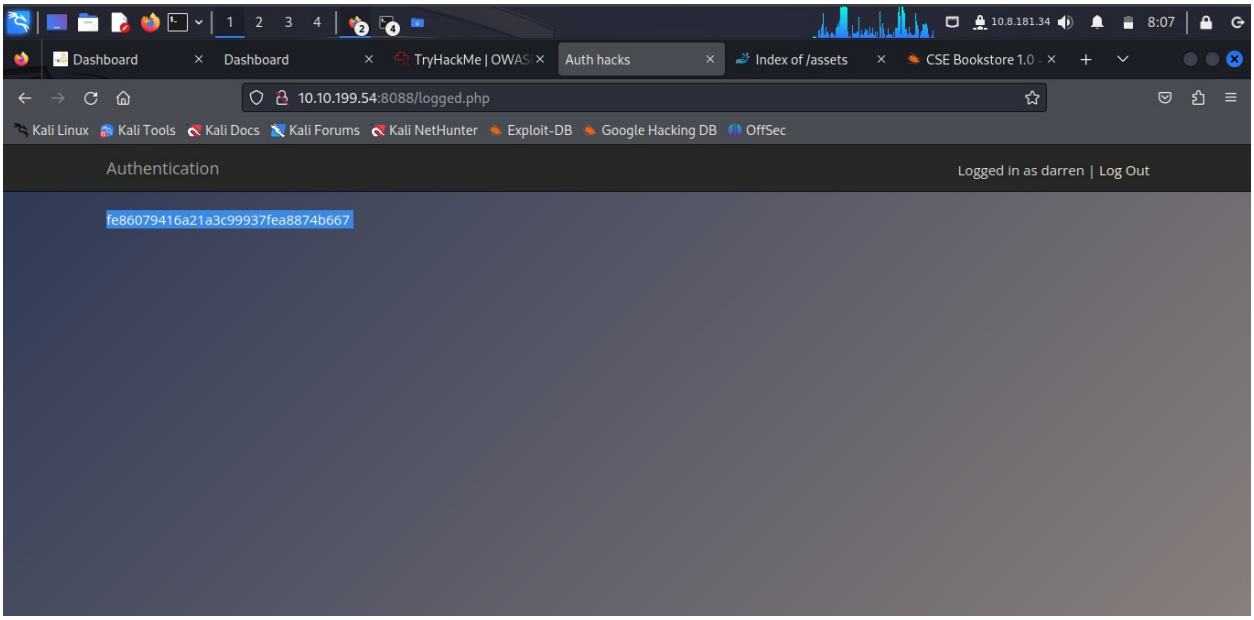
Brute force attacks: If a web application uses usernames and passwords, an attacker can try to launch brute force attacks that allow them to guess the username and passwords using multiple authentication attempts.

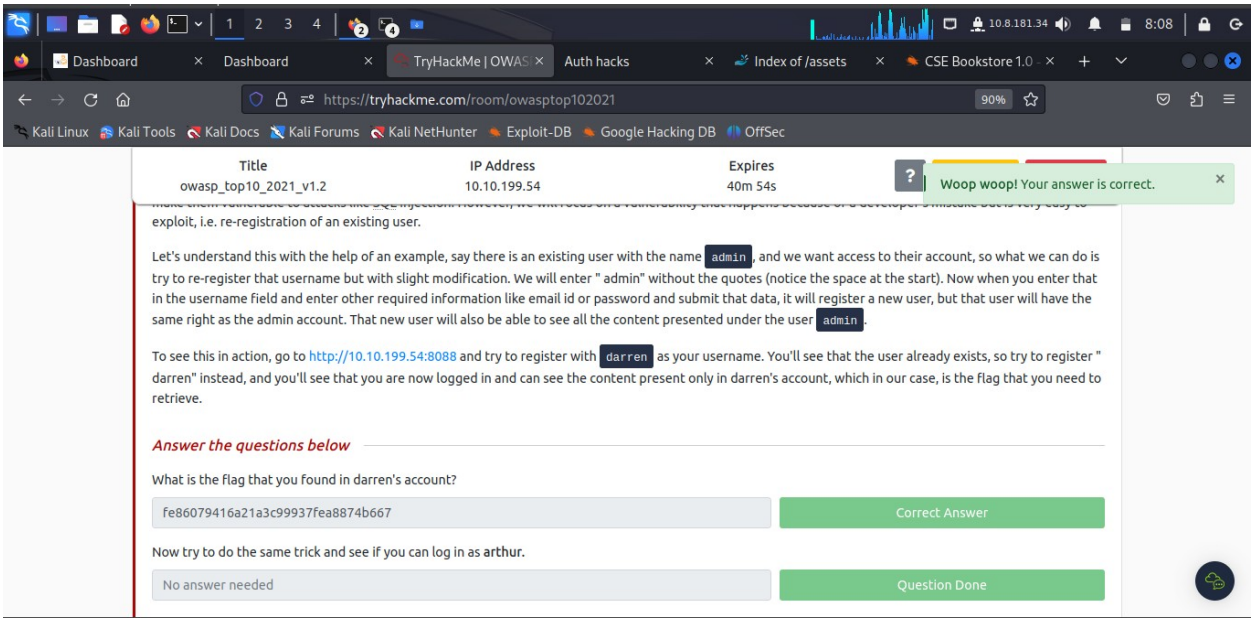
Use of weak credentials: Web applications should set strong password policies. If applications allow users to set passwords such as "password1" or common passwords, an attacker can easily guess them and access user accounts.

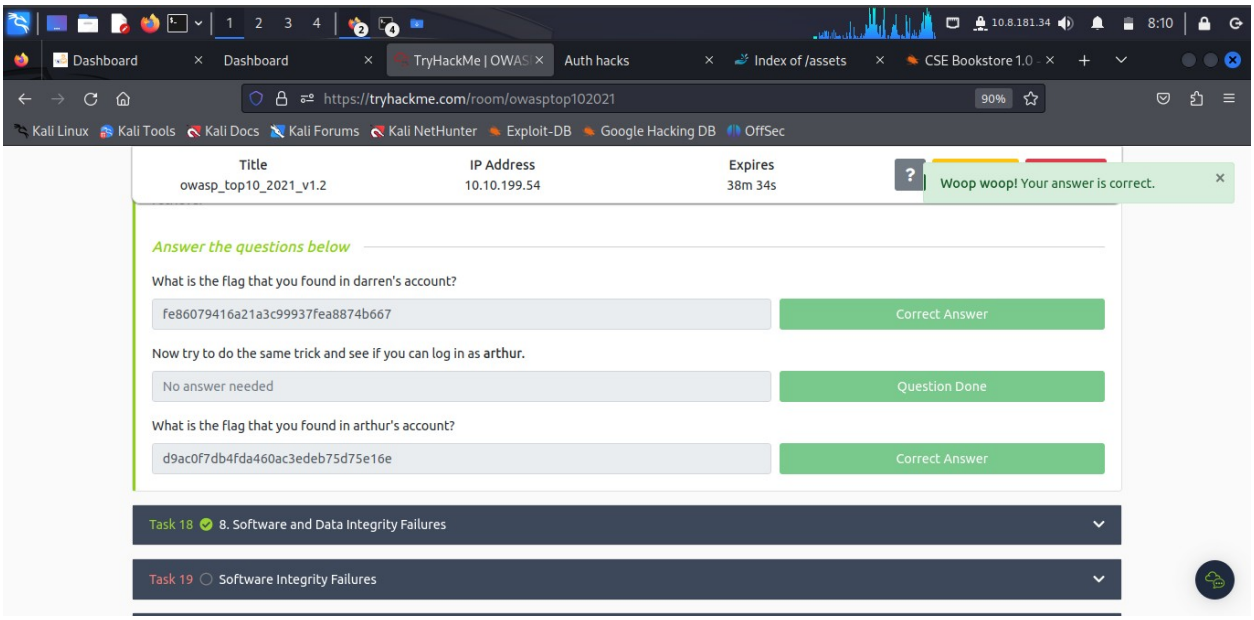
Weak Session Cookies: Session cookies are how the server keeps track of users. If session cookies contain predictable values, attackers can set their own session cookies and access users' accounts.

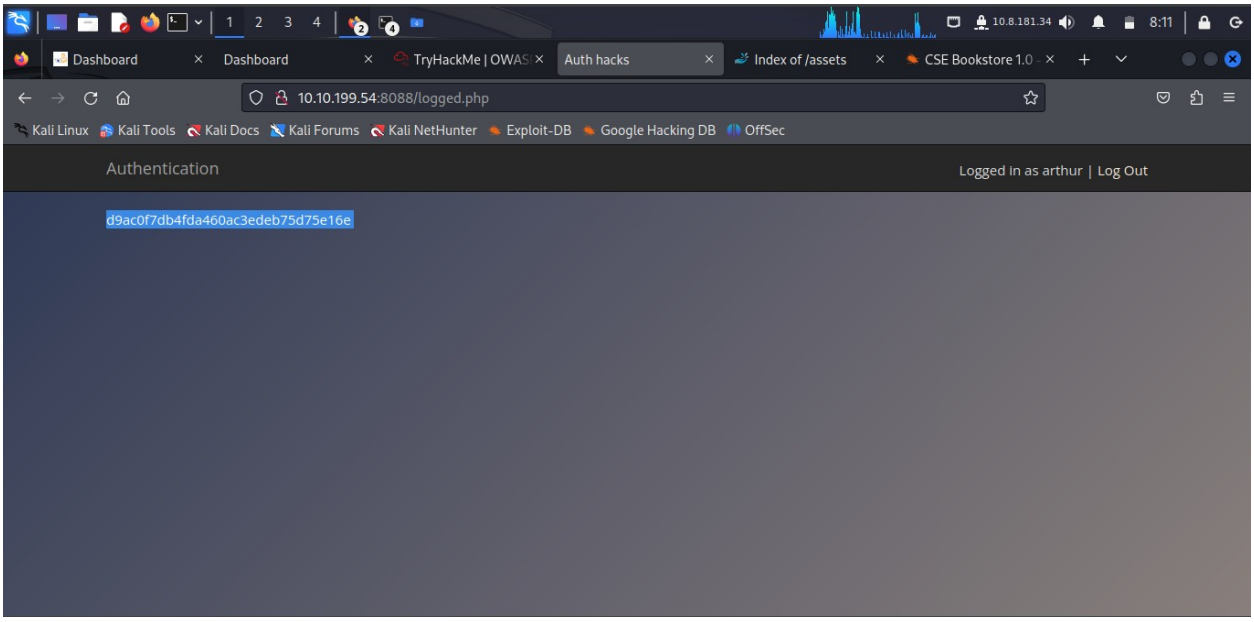
Task 17: Identification and Authentication Failures Practical

The learner looks at a logic flaw within the authentication mechanism.




Dashboard
Dashboard
TryHackMe | OWAS
Auth hacks
Index of /assets
CSE Bookstore 1.0
https://tryhackme.com/room/owasptop102021
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Title
owasp_top10_2021_v1.2
IP Address
10.10.199.54
Expires
40m 54s
Woop woop! Your answer is correct.
exploit, i.e. 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://10.10.199.54:8088 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.
Answer the questions below
What is the flag that you found in darren's account?
fe86079416a21a3c99937fea8874b667
Correct Answer
Now try to do the same trick and see if you can log in as arthur.
No answer needed
Question Done


Dashboard
Dashboard
TryHackMe | OWAS
Auth hacks
Index of /assets
CSE Bookstore 1.0
https://tryhackme.com/room/owasptop102021
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Title
owasp_top10_2021_v1.2
IP Address
10.10.199.54
Expires
38m 34s
Woop woop! Your answer is correct.
Answer the questions below
What is the flag that you found in darren's account?
fe86079416a21a3c99937fea8874b667
Correct Answer
Now try to do the same trick and see if you can log in as arthur.
No answer needed
Question Done
What is the flag that you found in arthur's account?
d9ac0f7db4fda460ac3edeb75d75e16e
Correct Answer
Task 18 8. Software and Data Integrity Failures
Task 19 Software Integrity Failures



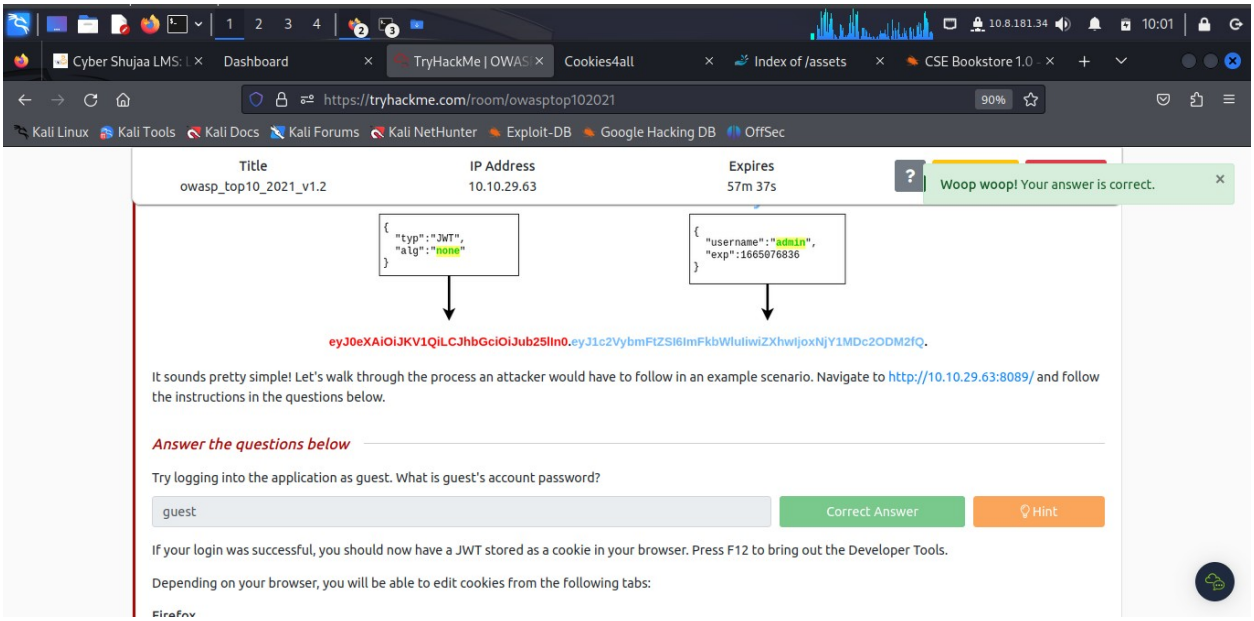
Task 18: 8. Software and Data Integrity Failures

Integrity refers to 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.

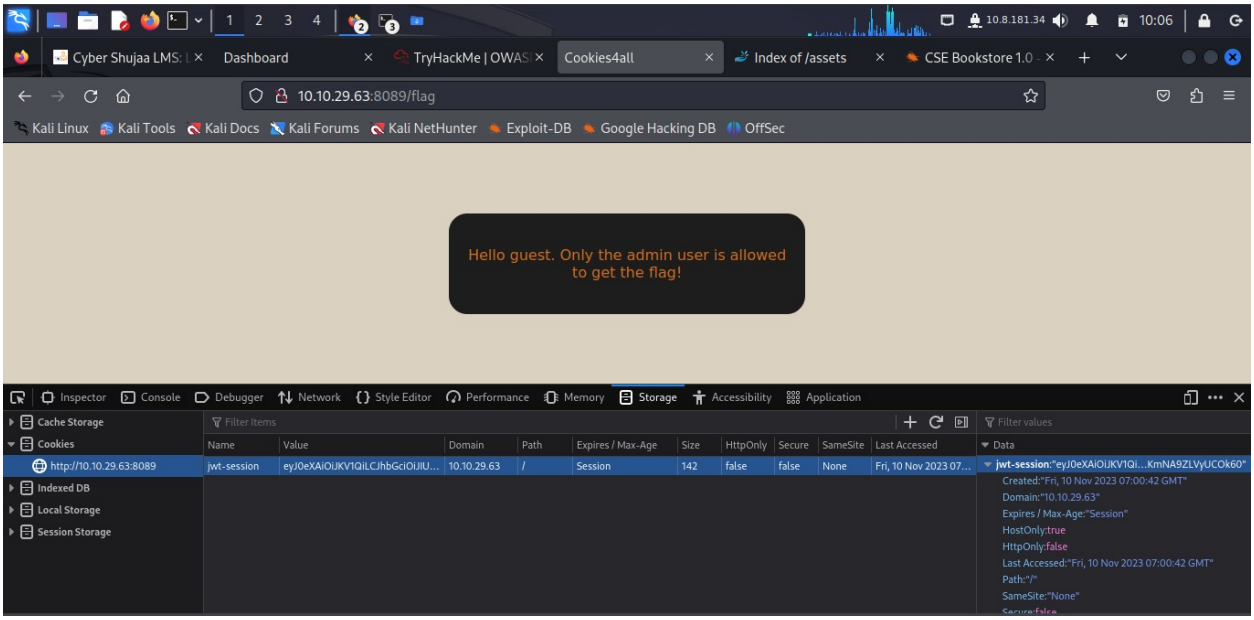
A **hash** sent alongside the file so that you can prove that the file you downloaded kept its integrity and wasn't modified in transit. A hash or digest is simply a number that results from applying a specific algorithm over a piece of data. When reading about hashing algorithms, you will often read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

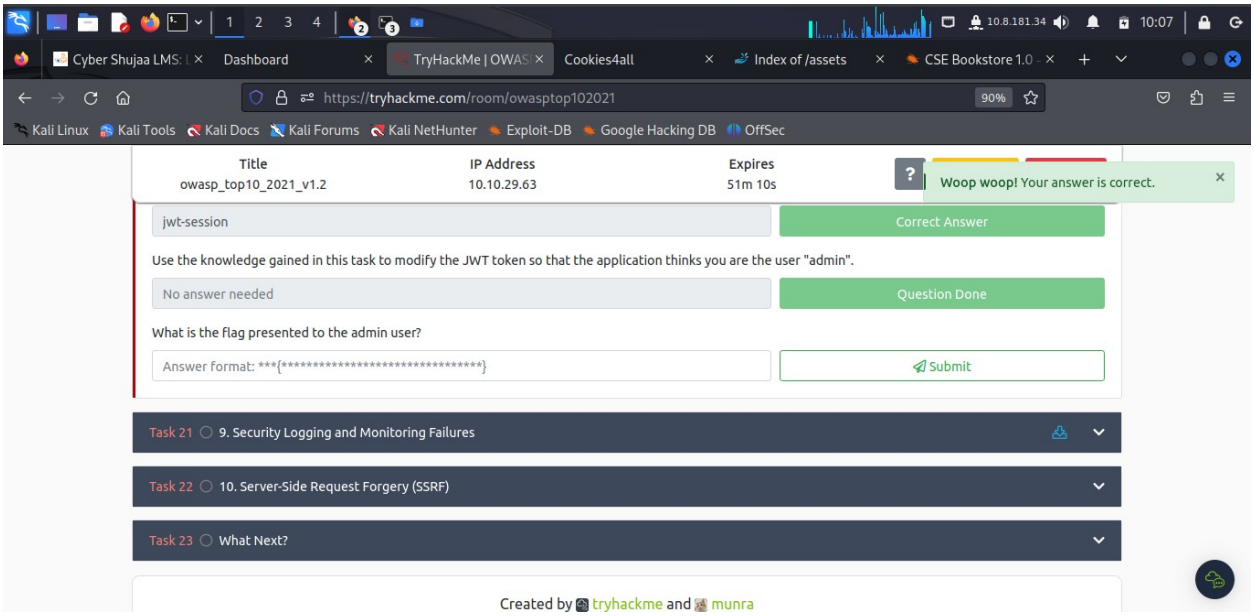
Task 19: Software Integrity Failures

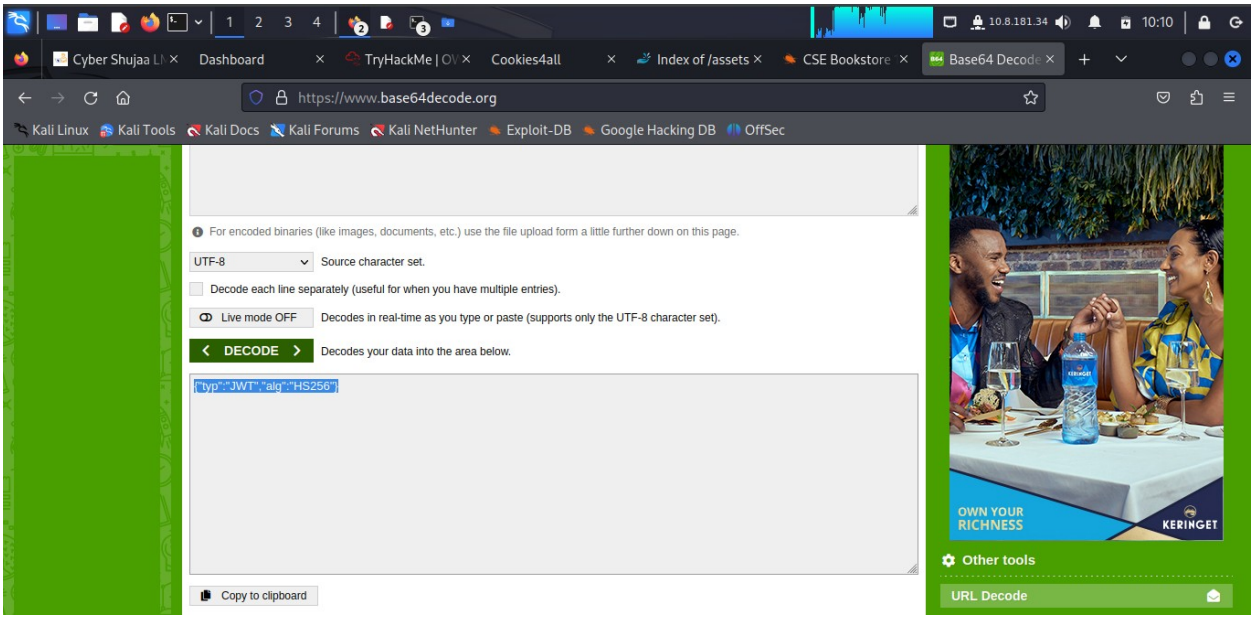
Task 20: Data Integrity Failures



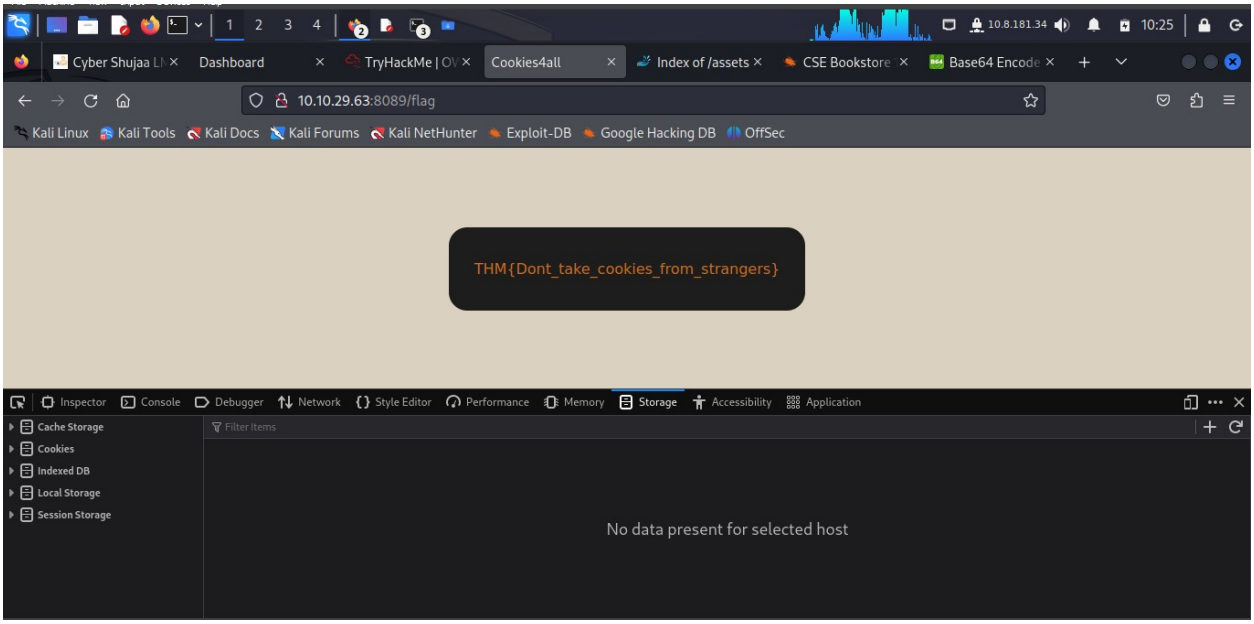
The learner found the password by hinting at the same username **guest**.

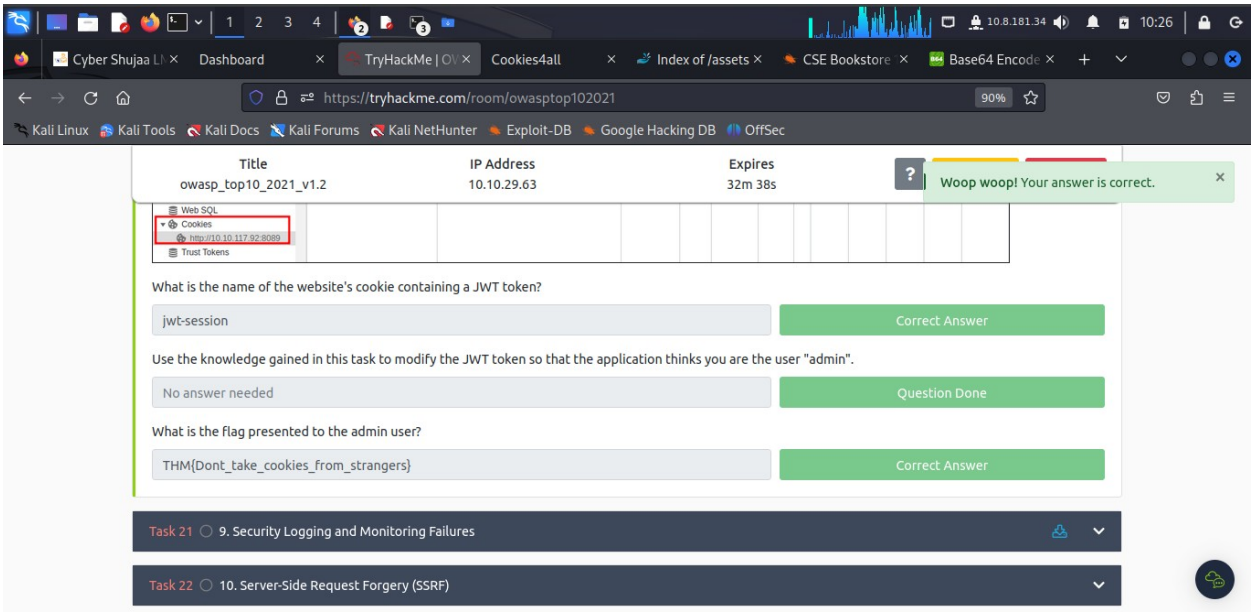






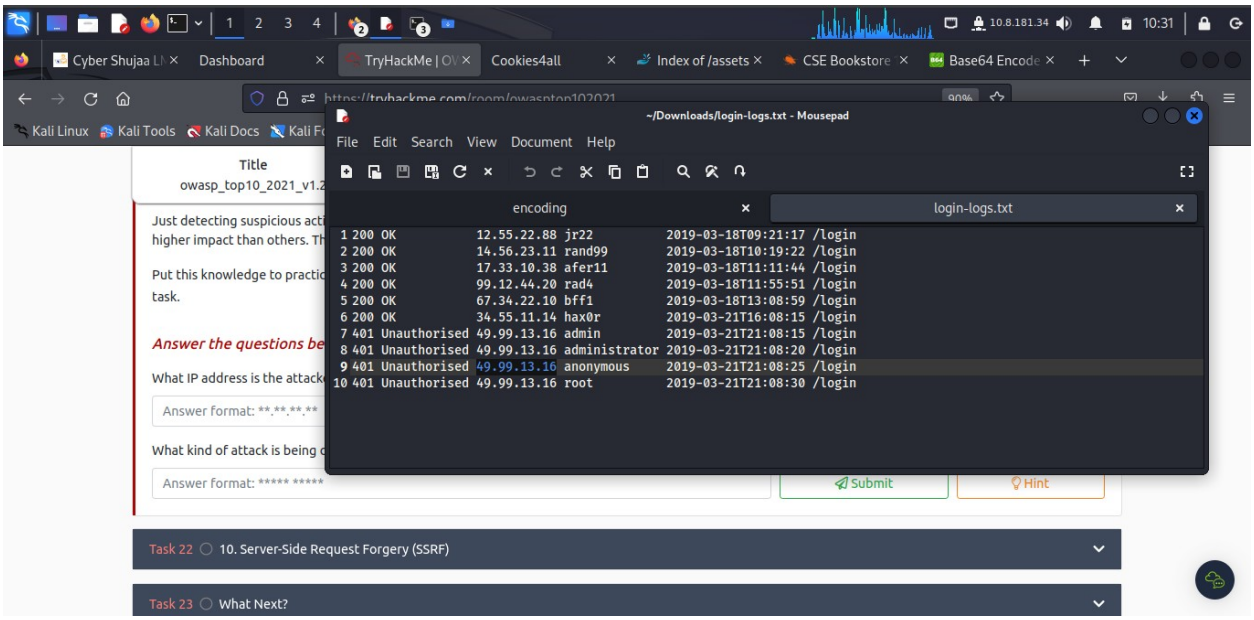
The learner decoded the first bit of the session token through base 64 encoder/decoder.

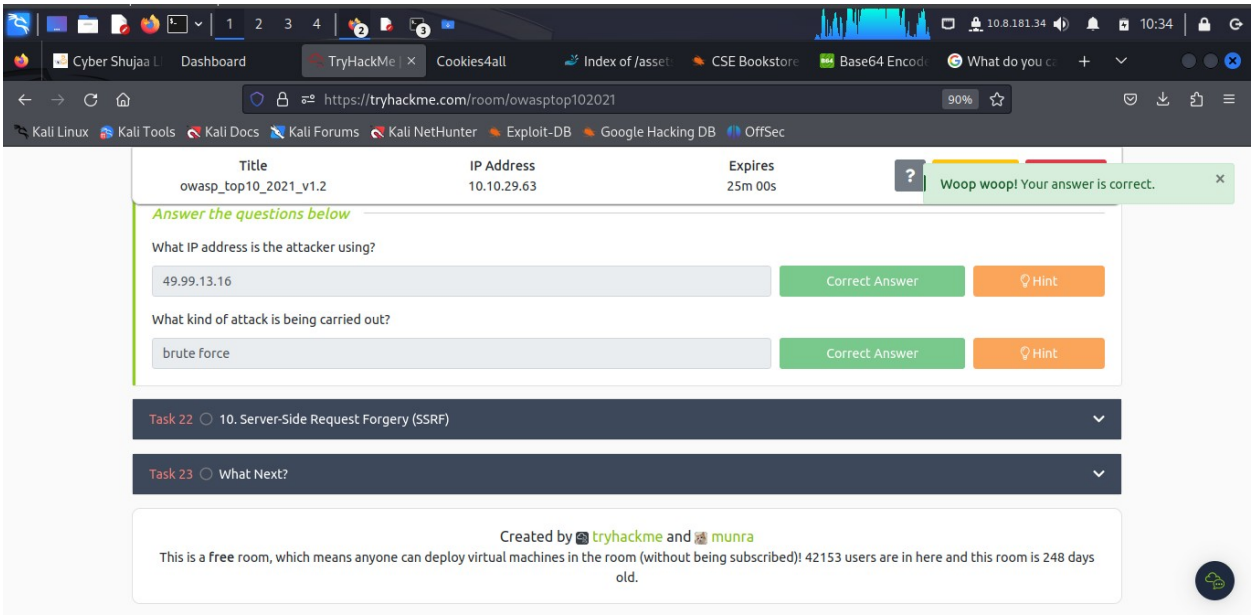

10.10.29.63:8089/flag
THM{Dont_take_cookies_from_strangers}
No data present for selected host


https://tryhackme.com/room/owasptop102021
Title
owasp_top10_2021_v1.2
IP Address
10.10.29.63
Expires
32m 38s
Woop woop! Your answer is correct.
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 22 10. Server-Side Request Forgery (SSRF)

Task 21: 9. Security Logging and Monitoring Failures

The learner has to know that 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.

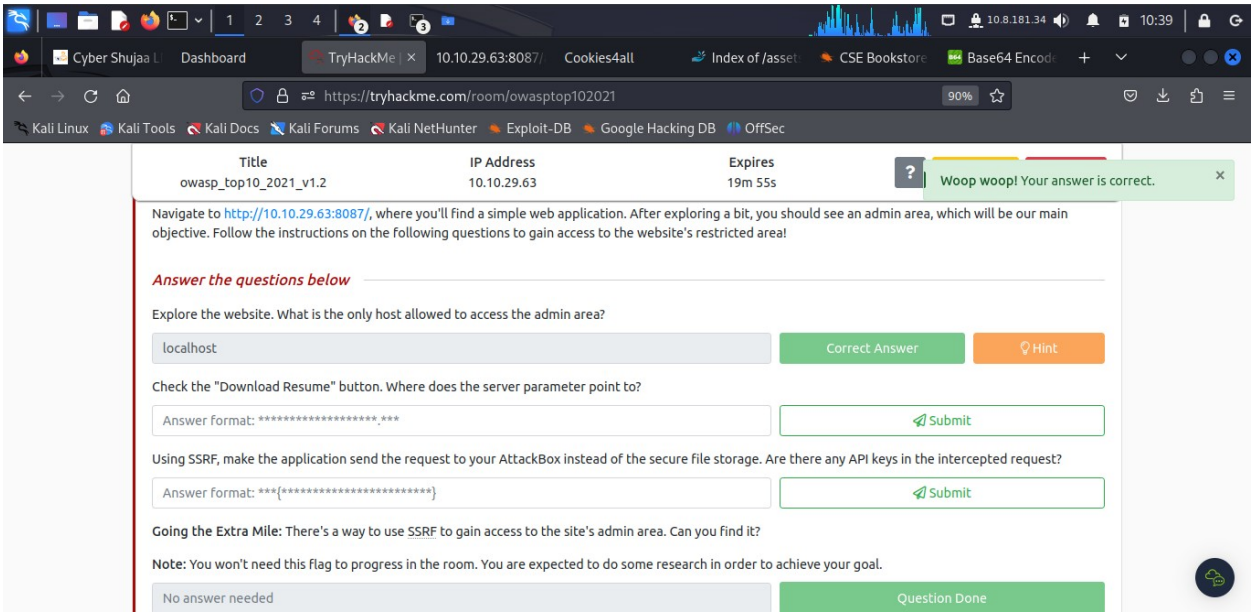




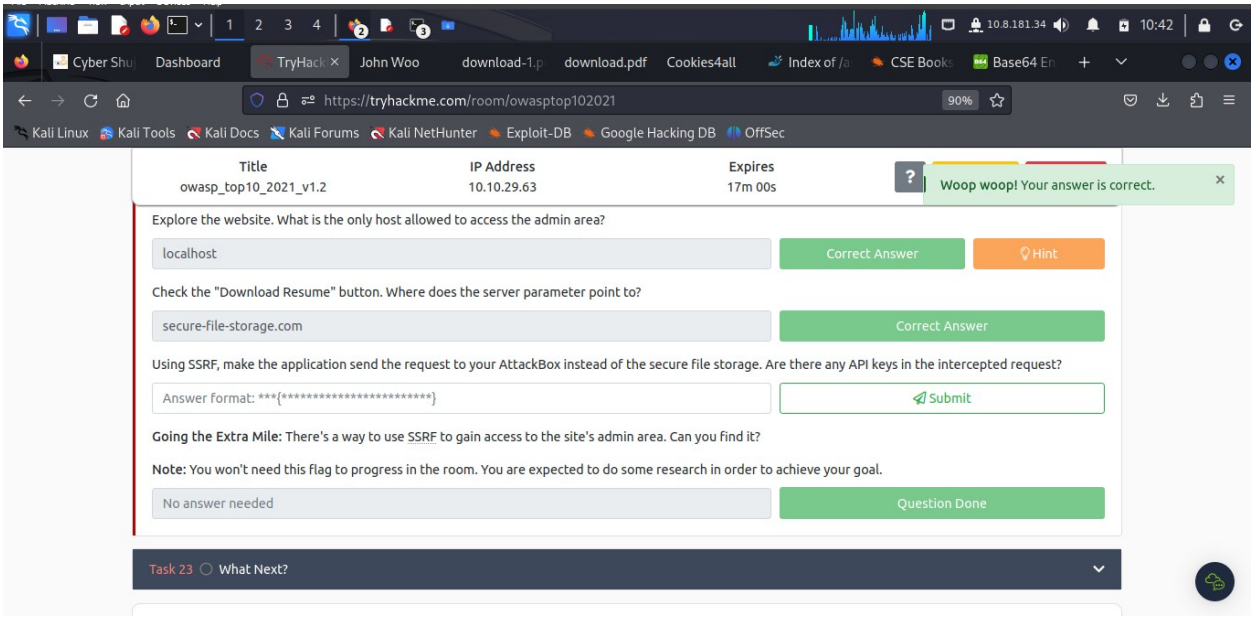
Task 22: 10. Server-Side Request Forgery (SSRF)

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.

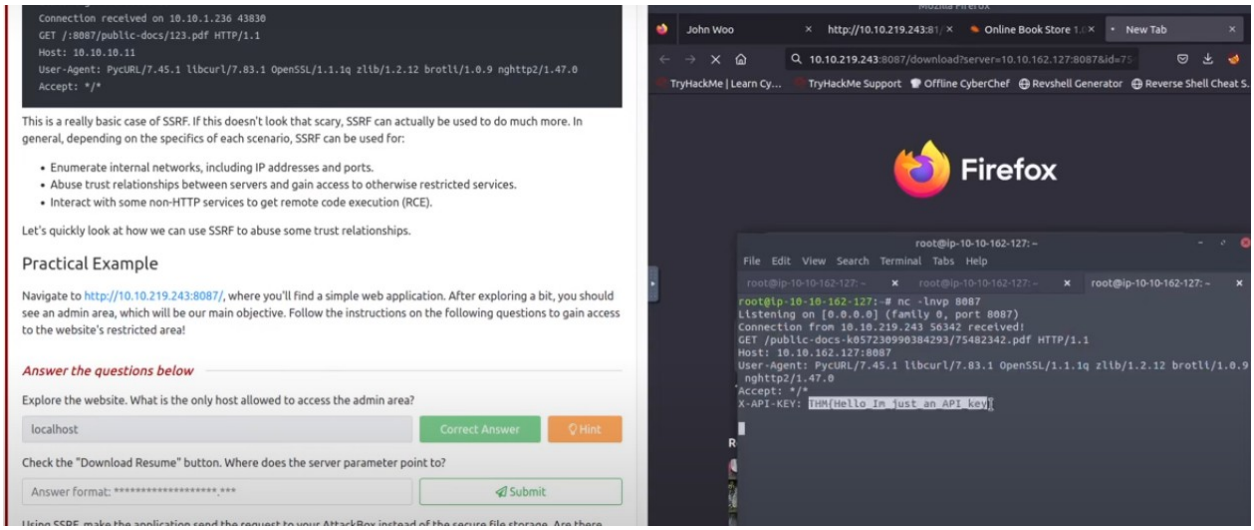
To know the only entity to access the admin is **localhost.**

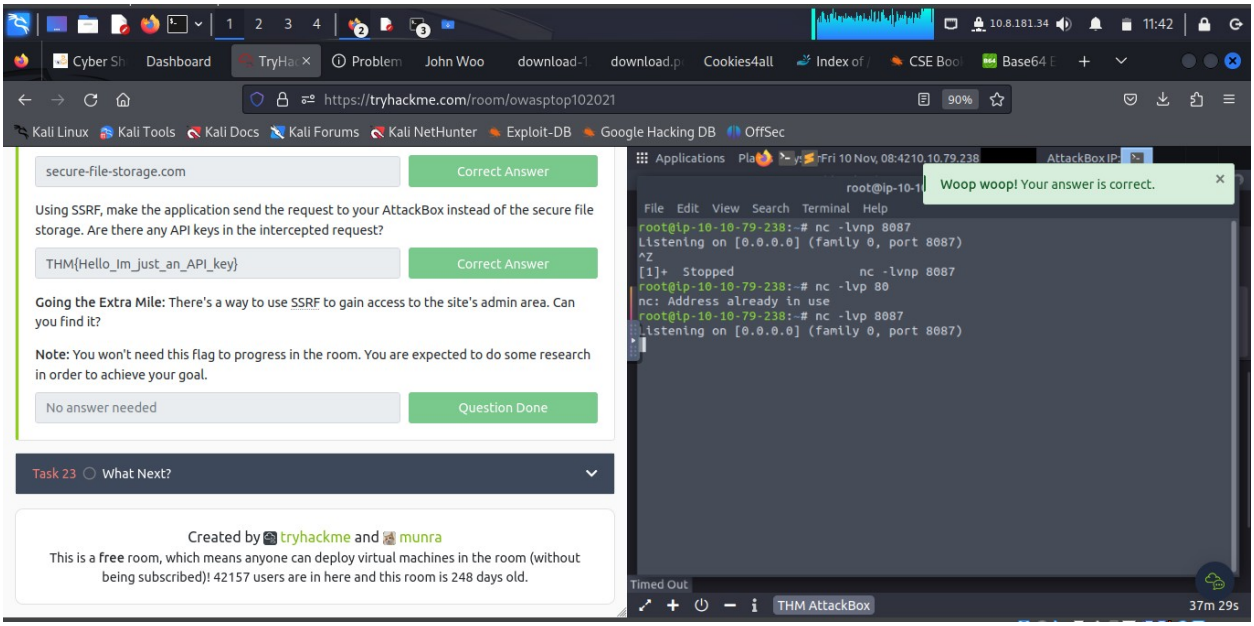


To get where the server points to, the learner hovered across the “Download Resume button” to find **secure-file-storage.com** as point to the server.

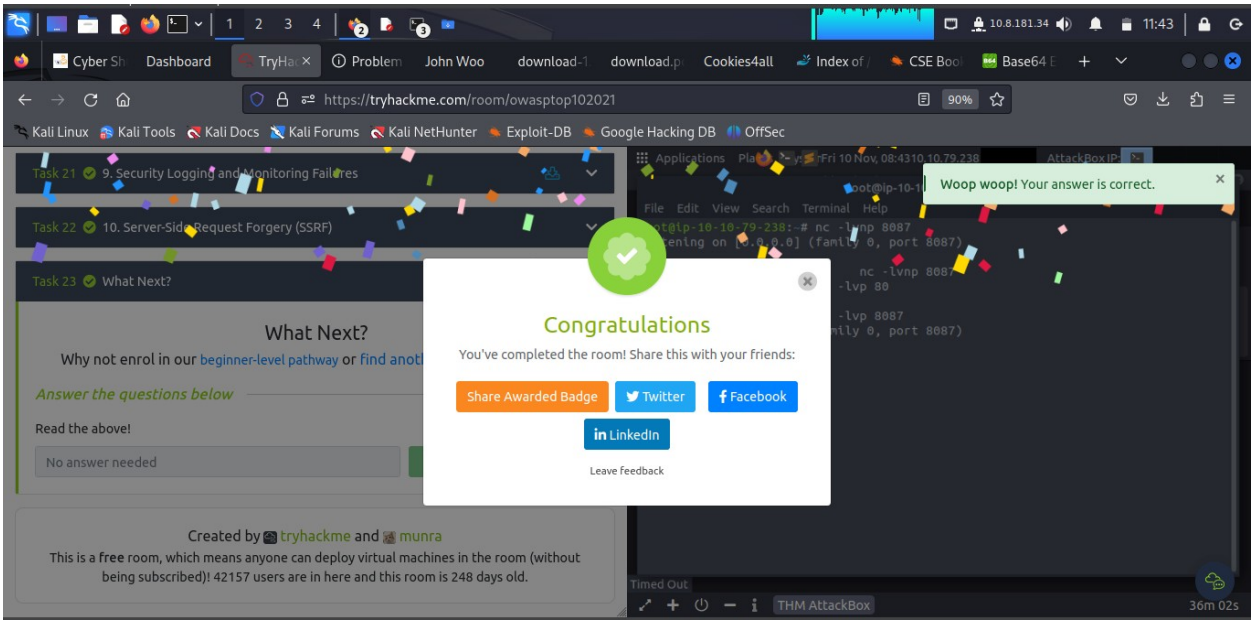


To make the application send the request to my AttackBox instead of the secure file storage, the learner assigned the **IP:10.10.162.127**





Task 23: What Next?



Conclusion

The learner now understands how to penetrate web applications, conduct research about sites to analyzed and overall OWASP skills necessary for a Security Analyst.

Completion link: https://tryhackme.com/room/owasptop102021