TRY HACK ME - OWASP 10

1. ***Lab Title : Insecure Direct Object Reference***

   Challenge Description:
   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. By object, we could mean a file, a user, a bank account in a banking application, or anything really.

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

   Questions:
   Look at other users' notes. What is the flag?
   → flag{fivefourthree}

2. ***Lab Title : Cryptographic Failure***

   Challenge Description:
   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.Cryptographic failures 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.

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

   Questions:
   What is the name of the mentioned directory?
   → /assets

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

Use the supporting material to access the sensitive data. What is the password hash of the admin user?
→ 6eea9b7ef19179a06954edd0f6c05ceb

Crack the hash.What is the admin's plaintext password?
→ qwertyuiop

Log in as the admin. What is the flag?
→ THM{Yzc2YjdkMjE5N2VjMzNhOTE3NjdiMjdl}

3. *Lab Title : Command Injection*

Challenge Description:
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 possibilities for the attacker from here are endless: they could list files, read their contents, run some basic commands to do some recon on the server or whatever they wanted, just as if they were sitting in front of the server and issuing commands directly into the command line. Once the attacker has a foothold on the web server, they can start the usual enumeration of your systems and look for ways to pivot around.

Vulnerability:
To complete the questions below, navigate to http://MACHINE_IP:82/ and exploit the cowsay server

Questions:
What strange text file is in the website's root directory?
→ drpepper.txt

How many non-root/non-service/non-daemon users are there?
→ 0

What user is this app running as?
→ apache

What is the user's shell set as?
→ /sbin/nologin

What version of Alpine Linux is running?
→ 3.16.0

4. ***Lab Title :Insecure Design***

Challenge Description:
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. Most of the time, these vulnerabilities occur when an improper threat modelling is made during the planning phases of the application and propagate all the way up to your final app. Some other times, insecure design vulnerabilities may also be introduced by developers while adding some "shortcuts" around the code to make their testing easier. 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.

Vulnerability:
Navigate to http://MACHINE_IP:85 and get into joseph's account. This application also has a design flaw in its password reset mechanism.

Questions:
What is the value of the flag in joseph's account?
→ THM{Not_3ven_c4tz_c0uld_sav3_U!}

5. ***Lab Title :Security Misconfiguration***

Challenge Description:
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.

Vulnerability:
Navigate to http://MACHINE_IP:86 and try to exploit the security misconfiguration to read the application's source code.

Questions:
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

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}

6. ***Lab Title :Vulnerable and Outdated Components***

Challenge Description:
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.

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

Questions:
What is the content of the /opt/flag.txt file?
→ THM{But_1ts_n0t_my_f4ult!}

7. ***Lab Title :Identification and Authentication Failures***

Challenge Description:
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.

Vulnerability:
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.

Questions:
What is the flag that you found in darren's account?
→ fe86079416a21a3c99937fea8874b667

What is the flag that you found in arthur's account?
→d9ac0f7db4fda460ac3edeb75d75e16e

8. ***Lab Title :Software and Data Integrity***

Challenge Description:
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.

Vulnerability:
You can go to https://www.srihash.org/ to generate hashes for any library .Navigate to http://MACHINE_IP:8089/ and follow the instructions in the questions.

Questions:
What is the SHA-256 hash of https://code.jquery.com/jquery-1.12.4.min.js?
→sha256-ZosEbRLbNQzLpnKIkEdrPv7lOy9C27hHQ+Xp8a4MxAQ=

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

What is the name of the website's cookie containing a JWT token?
→jwt-session

What is the flag presented to the admin user?
→THM{Dont_take_cookies_from_strangers}

9. *Lab Title :Security Logging and Monitoring Failures*

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

Vulnerability:
You can download it by clicking the Download Task Files button at the top of the task.

Questions:
What IP address is the attacker using?
→49.99.13.16

What kind of attack is being carried out?
→Brute Force

10. *Lab Title :Server Side Request Forgery*

Challenge Description:
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.

Vulnerability:
Navigate to http://MACHINE_IP:8087/, where you'll find a simple web application. After exploring a bit, you should see an admin area, which will be our main objective. Follow the instructions on the following questions to gain access to the website's restricted area!.

Questions:
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}