CYBR 350

Week # 10

Final

Abby Gopal

TryHackMe OWASP Top 10–2021

Task 4: Broken Access Control (IDOR Challenge)

1. Deploy the machine provided for the task.
2. Open a web browser and navigate to http://MACHINE\_IP.
3. You will be presented with a login page. Use the provided credentials to log in:
4. Username: noot
5. Password: test1234
6. After successfully logging in, you will likely be directed to a dashboard or home page.
7. Look for any functionality or option that allows you to view notes or data belonging to other users. This could be a feature like "View Notes" or "User Directory."
8. Access the feature or option that allows you to view other users' notes. This could involve clicking on a link or button labeled with the username of another user.
9. Explore the notes or data of other users to find the flag. The flag is typically a string of characters enclosed within THM{}.
10. Once you have found the flag, note it down.
11. Submit the flag as the final answer to complete the task.

A computer screen shot of a cloud

Description automatically generated

A computer screen with a screen showing a login and password

Description automatically generated

Task 8: Cryptographic Failures (Challenge)

1. Connect to the web application at http://MACHINE\_IP:81/ using a web browser.
2. On the login page of the web application, view the page source code by pressing CTRL + U or by right-clicking on the page and selecting "View Page Source."
3. Look through the page source code for any notes or hints left by the developer. Identify the mentioned directory where sensitive data is stored.
4. Note down the name of the mentioned directory. For example, if the directory is mentioned as /assets, then the answer would be /assets.
5. Navigate to the directory mentioned in the previous step. If the directory is /assets, append /assets/ to the URL in the browser's address bar and press Enter.
6. Explore the contents of the directory and identify any files that seem likely to contain sensitive data.
7. Identify the file that stands out as potentially containing sensitive data. For example, if the file is webapp.db, then note down this filename.
8. Download the webapp.db file from the directory.
9. Open a command-line interface and navigate to the directory where the webapp.db file is located.
10. Use the SQLite3 command-line tool to access the database by running the command sqlite3 webapp.db.
11. After accessing the SQLite3 prompt, list all tables in the database by running .tables.
12. Identify the table containing user information, which is typically named users. Retrieve the schema of the users table by running PRAGMA table\_info(users).
13. Look for the password hash of the admin user in the output. Note down the password hash.
14. Use a password-cracking tool or online service like CrackStation to crack the password hash and obtain the plaintext password of the admin user.
15. Log in to the web application using the admin credentials obtained in the previous step.
16. Once logged in, navigate to the flag section or access the flag using the admin privileges.
17. Note down the flag presented on the page, which typically starts with THM{}.
18. Submit the flag as the final answer to complete the task.

A computer screen with a login box

Description automatically generatedA computer screen shot of a computer

Description automatically generated

A computer screen with a screen on

Description automatically generatedA computer screen with a password

Description automatically generatedA computer screen with a screen showing a program

Description automatically generated with medium confidence

Task 10: Command Injection

1. Access the provided link http://machine-ip:82/ in your web browser.
2. Start by attempting a command injection using the $(whoami) command to determine the current user.
3. Note down the result, which is likely to be apache.
4. Use the command $(ls) to list the files in the root directory.
5. Identify any unusual files in the root directory, such as drpepper.txt.
6. Next, execute the command $(cat /etc/passwd) to list all users on the system.
7. Analyze the output to identify non-root/non-service/non-daemon users. If none are found, the answer to the question about the number of such users is 0.
8. Identify the user's shell by inspecting the /etc/passwd file. Look for the user apache and note down its shell, which may be /sbin/nologin.
9. Execute the command $(cat /etc/os-release) to determine the version of Alpine Linux running on the system.
10. Note down the version number, which is likely to be something like 3.16.0.
11. Submit the answers to the questions posed in the task:
12. Strange text file in the website's root directory: drpepper.txt
13. Number of non-root/non-service/non-daemon users: 0
14. User's shell set as: apache
15. User's shell set as: /sbin/nologin
16. Version of Alpine Linux running: 3.16.0

A computer screen with a black background

Description automatically generatedA computer screen shot of a computer screen

Description automatically generatedA computer screen with a login page

Description automatically generatedA computer screen with a login box

Description automatically generated

A computer screen with a box and text on it

Description automatically generatedA computer screen with a blue and white text

Description automatically generatedA computer screen with a blue and white text

Description automatically generated

Task 11: Insecure Design

1. Boot up the virtual machine (VM) and open Firefox.
2. Navigate to http://MACHINE\_IP:85 in the web browser.
3. On the login page, observe that it requires a username and password.
4. Since the task mentions a person named Joseph and a potential password reset mechanism, click on the "I forgot my password…" button.
5. The system will prompt for the username. Enter "joseph" as the username and proceed.
6. The hint provided suggests guessing answers to security questions. As indicated, try guessing common colors, such as Red, Orange, Yellow, Green, Blue, Indigo, and Violet (ROYGBIV).
7. The system accepts the correct answer. Note down the automatically generated temporary password provided by the system.
8. Copy or write down the temporary password and use it to log in to Joseph's account.
9. Once logged in, navigate to Flag.txt or any relevant section to retrieve the flag.
10. There is no need to submit an answer as the task does not require it. The value of the flag obtained from Joseph's account is THM{Not\_3ven\_c4tz\_c0uld\_sav3\_U!}.

A computer screen with a white and red text

Description automatically generatedA computer screen with a mouse pointer

Description automatically generatedA computer screen with a screen on

Description automatically generated

Task 12: Security Misconfiguration

1. Open a web browser and navigate to http://MACHINE\_IP:86/console to access the Werkzeug console.
2. Once on the Werkzeug console page, locate the input area where Python code can be entered.
3. Input the provided Python code to execute the ls -l command on the server:

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

1. Execute the code by pressing Enter or clicking on the appropriate button to run the script.
2. Review the output returned by the code. Look for a file with the .db extension, as this is the database file.
3. Note down the name of the database file, which is the answer to the first question.
4. Next, modify the Python code to read the contents of the app.py file. Replace the existing code with the following import os; print(os.popen("cat app.py").read())
5. Execute the modified code by pressing Enter or clicking on the appropriate button.
6. Review the output returned by the code. Look for the value of the secret\_flag variable in the source code of app.py.
7. Note down the value of the secret\_flag variable, which is the answer to the second question.
8. Submit the answers to both questions in the appropriate answer format provided in the task.

A computer screen with a computer screen

Description automatically generatedA computer screen with a computer screen

Description automatically generated

Task 15: Vulnerable and Outdated Components

1. Open Firefox and navigate to the given website provided in the task.
2. Upon inspection, identify that the website is a bookstore application.
3. Visit the Exploit Database website and search for recent unauthenticated remote code execution (RCE) exploits related to bookstore applications.
4. Download the exploit file named "47887.py" from the Exploit Database.
5. Open a terminal window and navigate to the directory where the exploit file is downloaded.
6. Execute the exploit script using Python 3 by running the command python3 47887.py [insert URL]. Replace [insert URL] with the URL of the vulnerable bookstore application.
7. When prompted, type "y" to launch the shell.
8. Once the shell is launched, use the command cat /opt/flag.txt to reveal the content of the flag file.
9. The content of the flag file will be displayed in the terminal.
10. Capture the content of the flag file, which is the answer to the task.
11. Submit the flag content in the appropriate answer format provided in the task.

A computer screen with a red and white text

Description automatically generated

Task 17: Identification and Authentication Failures Practical

1. Open a web browser and navigate to http://MACHINE\_IP:8088.
2. Try to register with the username "darren". You'll receive a message indicating that the user already exists.
3. Instead, try registering with the username " darren" (with a space before the username).
4. After registering, you'll receive a message indicating success, but the username will be " darren" with the space.
5. Log in using the credentials you provided during registration.
6. Once logged in, you'll have access to darren's account, including any content restricted to that account, such as the flag.
7. Retrieve the flag from darren's account.
8. Repeat the same method to access the username "arthur" by registering with " arthur" (with a space before the username).
9. Log in using the credentials you provided during registration.
10. Once logged in, you'll have access to arthur's account, including any content restricted to that account, such as the flag.
11. Retrieve the flag from arthur's account.

A computer screen with a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generatedA computer screen with a black box

Description automatically generatedA computer screen shot of a computer

Description automatically generated

Task 19: Software Integrity Failures Security Misconfiguration

1. Open a new browser tab or use the attack box and navigate to https://www.srihash.org/.
2. In the input field provided on the website, enter the given URL: https://code.jquery.com/jquery-1.12.4.min.js.
3. Ensure that you select SHA-256 as the hashing algorithm.
4. Click on the "Hash!" button or equivalent to calculate the integrity hash for the specified URL.
5. Once the hash is generated, it should be displayed on the website.
6. Compare the generated SHA-256 hash with the expected hash provided in the task.
7. If the hashes match, record the generated SHA-256 hash as your answer.
8. The answer format should be in the form of SHA256-[generated hash], for example, SHA256-ZosEbRLbNQzLpnKIkEdrPv7lOy9C27hHQ+Xp8a4MxAQ=.

A computer screen with text and a black arrow

Description automatically generated

Task 20: Data Integrity Failures

1. Start your machine and the attack box, then navigate to the provided website at http://MACHINE\_IP:8089/.
2. Attempt to log into the guest account with a random password. You'll receive a message providing instructions on how to proceed.
3. Once logged in as a guest, you'll receive another message.
4. Press SHIFT+F12 to bring up the Developer Tools in your browser.
5. Use the "head" and "payload" parts of the JWT token. The header is everything up to the first ".", and the payload is everything between the first and second ".".
6. Decode the header and payload using a base64 encoder-decoder.
7. Change the algorithm from "HS256" to "none" in the header and change the username from "guest" to "admin" in the payload.
8. Encode the modified header and payload separately.
9. Combine the modified header and payload and paste the resulting JWT token into the value of the cookie.
10. Refresh the page to execute the modified JWT token.
11. As a reminder, the guest's account password is "guest".
12. After a successful login, you'll have a JWT token stored as a cookie in your browser. Access the Developer Tools again.
13. Edit the cookie containing the JWT token to make the application think you are the user "admin".
14. Upon successfully modifying the JWT token, the flag will be presented to the admin user. The format for the flag is THM{Dont\_take\_cookies\_from\_strangers}.

A computer screen with text on it

Description automatically generatedA computer screen with a sign on it

Description automatically generatedA computer screen with a black and orange text

Description automatically generatedA computer screen shot of a white square

Description automatically generatedA computer screen with a message on it

Description automatically generated

Task 21: Security Logging and Monitoring Failures

1. Observe the IP address that appears repeatedly within a 15-second timeframe. This IP address indicates the attacker's location.
2. Identify the IP address used by the attacker, which is 49.99.13.16.
3. Notice the pattern of the attacker attempting to log in with different usernames within a short timeframe.
4. Recognize that the repeated attempts with different usernames suggest a brute-force attack.
5. Understand that a brute force attack involves trying various combinations of usernames and passwords to gain unauthorized access to user accounts.
6. Confirm that the type of attack being carried out is indeed a brute force attack, given the repeated login attempts with different usernames.

A screenshot of a computer

Description automatically generated

Task 22: Server-Side Request Forgery (SSRF)

1. Start the provided virtual machine and the attack box.
2. Open Firefox and navigate to http://MACHINE\_IP:8087/.
3. Attempt to access the Admin Area, resulting in a denial message due to access restrictions.
4. Note the message indicating denial due to not being "localhost."
5. Identify the server parameter value as secure-file-storage.com, which is the target to be manipulated.
6. Access the provided link resembling http://10.10.182.198:8087/download?server=secure-file-storage.com:8087&id=75482342.
7. Replace secure-file-storage.com with the IP address of your attack box.
8. Open a terminal and execute nc -lvnp 8087 to listen for incoming connections.
9. Use the altered link with the modified server parameter.
10. Monitor the terminal for any incoming connections, indicating successful interception.
11. Modify the intercepted request's server parameter to http://localhost:8087/admin%23&id=75482342, ensuring access to the Admin Area.
12. Create a new URL with the modified server parameter, like http://10.10.190.86:8087/download?server=http://localhost:8087/admin%23&id=75482342.
13. Access this URL to explore the Admin Area.
14. Verify that only localhost can access the Admin Area, as deduced from the initial access attempt and confirmed through successful exploitation.
15. Investigate the "Download Resume" button to determine where the server parameter points.
16. Identify that the server parameter points to secure-file-storage.com.
17. Examine the intercepted request for any API keys or sensitive information.
18. Retrieve the flag containing the API key from the intercepted request, confirming the successful exploitation of SSRF vulnerabilities.

A computer screen with text on it

Description automatically generatedA computer screen with a black and white screen

Description automatically generated

A screenshot of a computer

Description automatically generated

The TryHackMe OWASP Top 10 room offers an interactive learning experience that bridges

theoretical knowledge of web application vulnerabilities with practical hands-on exercises. Each activity corresponds closely with the theoretical aspect of a specific vulnerability category, such as Broken Access Control, Cryptographic Failures, Injection, and more. In the Broken Access Control category, the theoretical aspect emphasizes the significance of implementing proper access controls to prevent unauthorized access to resources. Participants learn about the risks of inadequate enforcement and how attackers can exploit these weaknesses to gain unauthorized access or perform actions reserved for privileged users. Practical exercises in this category demonstrate various techniques to bypass access controls, access unauthorized resources, or escalate privileges, thereby reinforcing the importance of robust access control mechanisms.

Cryptographic Failures focus on theoretical knowledge regarding encryption, hashing, and other cryptographic techniques used to secure data in web applications. Participants learn about the importance of cryptographic solid practices and the risks of cryptographic weaknesses. Hands-on activities involve exploiting vulnerabilities in cryptographic implementations, such as insecure storage of sensitive data or weak encryption algorithms, to demonstrate how attackers can compromise the confidentiality and integrity of data. Injection vulnerabilities, including SQL injection, command injection, and cross-site scripting (XSS), are covered in the Injection category. Theoretical understanding encompasses the risks posed by inadequate input validation and sanitization, which can allow attackers to inject and execute arbitrary code within the application context. Practical exercises demonstrate how attackers can exploit injection vulnerabilities to manipulate databases, execute unauthorized commands, or steal sensitive information, underscoring the importance of robust input validation mechanisms. In the Insecure Design category, participants learn about flaws in the design of web applications, such as improper authentication mechanisms or inadequate session management. Theoretical aspects highlight how design flaws can lead to unauthorized access, data leakage, or privilege escalation. Practical activities provide hands-on experience exploiting design flaws to bypass authentication mechanisms, hijack sessions, or escalate privileges, emphasizing the importance of secure design principles.

Security Misconfiguration focuses on theoretical knowledge regarding the importance of secure configuration settings and the risks of misconfigurations. Participants learn about common misconfiguration pitfalls and their potential impact on application security. Practical exercises demonstrate how misconfigured settings can expose sensitive data, allow unauthorized access, or compromise the integrity of the application, reinforcing the importance of thorough configuration management practices.

In a real-world system, these categories' vulnerabilities can have severe consequences. For instance, inadequate access controls or authentication mechanisms can lead to data breaches, unauthorized access to sensitive information, financial losses, and damage to reputation. Similarly, vulnerabilities such as injection attacks, cryptographic failures, or insecure configurations can result in manipulation of data, compromise of system integrity, and exploitation by malicious actors. These risks underscore the critical importance of implementing robust security measures and proactive mitigation strategies to safeguard web applications and their users.

Recommended mitigation strategies, based on OWASP guidelines and other credible sources, include implementing proper access controls and authentication mechanisms to enforce least privilege, conducting regular security assessments and patch management, implementing secure coding practices, configuring secure settings and hardening servers, and implementing monitoring and logging mechanisms to detect and respond to security incidents promptly. Additionally, providing comprehensive security awareness training for developers, administrators, and users is essential to ensure a holistic approach to web application security.