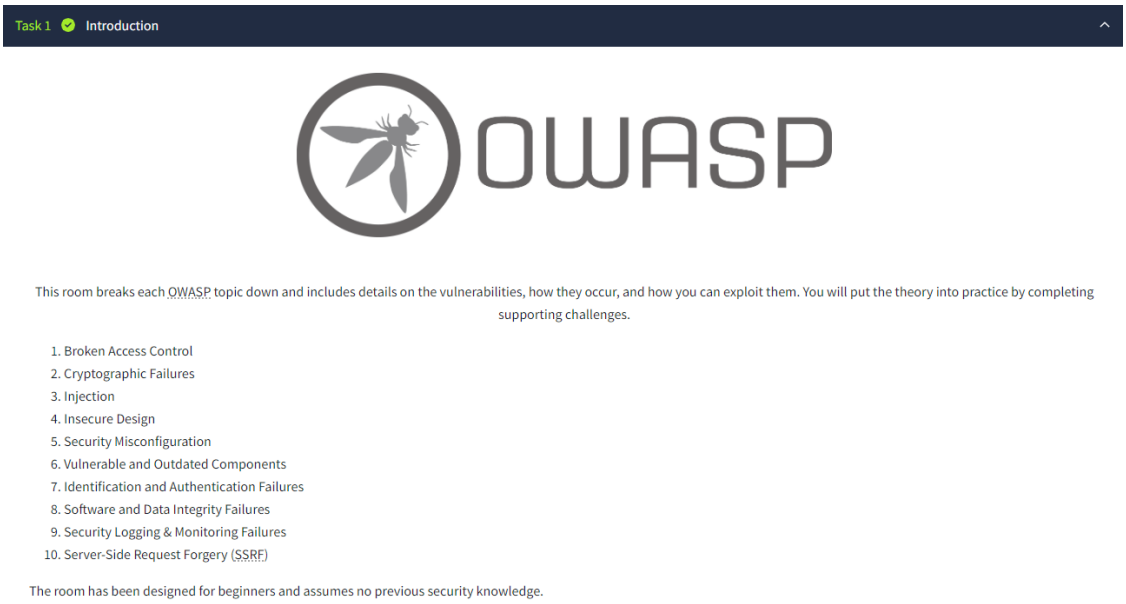
OWAPS-TOP-10-2021



1.Broken Access Control

Broken Access Control occurs when users can perform actions beyond their intended permissions. This can lead to unauthorized access to sensitive data, modification of user data, or even administrative actions without proper authorization. Ensuring robust access controls is vital to maintaining the security and integrity of applications.

Key Points:

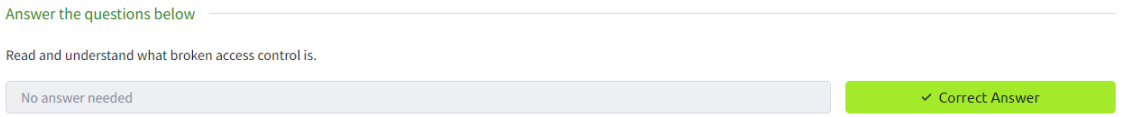
- Unauthorized Access: Users gaining access to data or functions they should not be able to.
- Privilege Escalation: Attackers gaining higher privileges than intended, potentially compromising the system.
- Insecure Direct Object References (IDOR): Accessing unauthorized resources through manipulated URLs or identifiers.
- Access Control Misconfiguration: Incorrect implementation or configuration of access control mechanisms.

Mitigation Strategies:

- Implement Principle of Least Privilege: Ensure users have the minimum level of access necessary to perform their tasks.

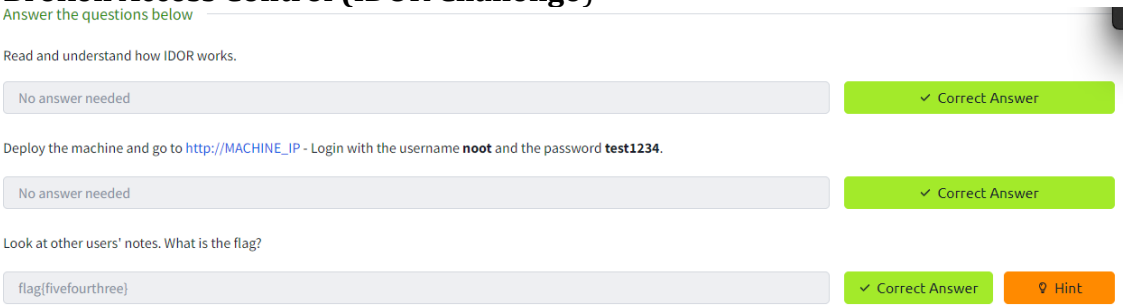
- Regular Access Control Reviews: Periodically review and update access controls to ensure they are correctly applied.
- Use Secure Access Control Mechanisms: Implement robust access control mechanisms such as role-based access control (RBAC) and attribute-based access control (ABAC).
- Conduct Regular Audits: Regularly audit access logs and permissions to detect and address potential security issues.
- Broken Access Control highlights the need for meticulous implementation and regular review of access control measures to prevent unauthorized actions and protect sensitive data.

Question:



Question 1: Question 1: Try to reset joseph's password. Keep in mind the method used by the site to validate if you are indeed joseph. **Answer:** No Need to Answer

Broken Access Control (IDOR Challenge)



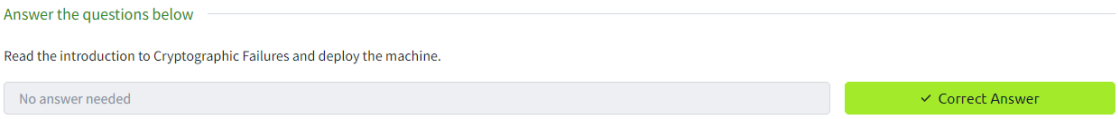
2.Cryptographic Failures

Cryptographic Failures refer to the improper implementation or use of cryptographic algorithms and protocols, which can lead to the exposure of sensitive data. These failures are critical because they undermine the integrity, confidentiality, and authenticity of data. Common issues include weak encryption algorithms, poor key management practices, and the improper handling of cryptographic processes.

Key Points:

- Weak Algorithms: Using outdated or weak encryption algorithms that can be easily broken by attackers.
- Improper Key Management: Failing to securely generate, store, and manage cryptographic keys, leading to potential unauthorized access.
- Insufficient Protection: Not adequately protecting sensitive data during transmission and storage, making it vulnerable to interception and theft.
- Misuse of Cryptographic Functions: Incorrectly using cryptographic functions, such as failing to use salt with hashing algorithms or not using encryption when needed. Mitigation Strategies:
- Use Strong Encryption: Implement strong, industry-standard encryption algorithms to protect sensitive data.
- Key Management Best Practices: Follow best practices for key management, including secure generation, storage, and rotation of cryptographic keys.
- Protect Data in Transit and at Rest: Ensure that sensitive data is encrypted both during transmission and when stored, using robust cryptographic protocols.
- Regular Audits and Updates: Conduct regular security audits and updates to ensure that cryptographic implementations remain secure and up-to-date with current standards.
- Cryptographic Failures underscore the importance of properly implementing and managing cryptographic mechanisms to ensure the security and privacy of data within applications.

Question:



Question 1: Read the introduction to Cryptographic Failures and deploy the machine.
Answer: No Need to Answer

3.Injection

Injection flaws are very common in applications today. These 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.

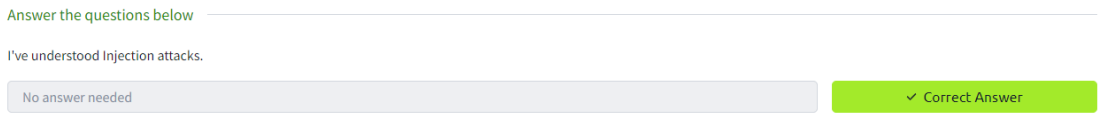
- Command Injection: This occurs when user input is passed to system commands. As a result, an attacker can execute arbitrary system commands on application servers, potentially allowing them to access users' systems. The main defence for preventing injection attacks is ensuring that user-controlled 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.

- Dangerous characters or input is classified as any input that can change how the underlying data is processed. Instead of manually constructing allow lists or stripping input, various libraries exist that can perform these actions for you.

Question:



Question 1: I've understood Injection attacks. **Answer:** No Need to Answer

4.Insecure Design

Insecure Design refers to the flaws and vulnerabilities that occur during the development phase of an application. These design flaws can arise due to inadequate attention to security requirements or a lack of understanding of potential threats. Insecure design issues are fundamentally different from insecure implementations, which result from mistakes in coding practices.

Key Points:

- Lack of Security Controls: Absence of necessary security controls, such as authentication, authorization, and input validation, can leave applications vulnerable to attacks.

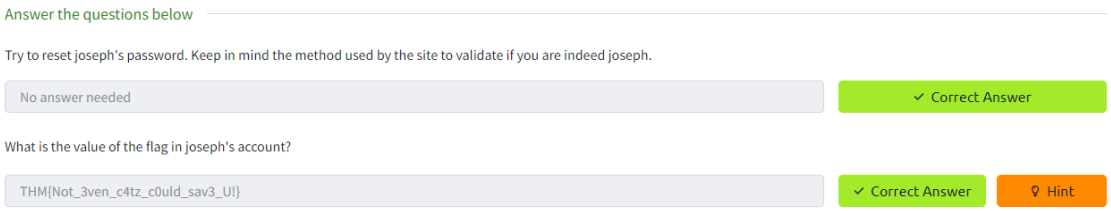
- Misunderstanding of Security Requirements: Insufficient understanding or documentation of security requirements during the design phase.

- Failure to Anticipate Threats: Not considering potential threats and attack vectors during the application design process.
- Inadequate Use of Secure Design Patterns: Overlooking best practices and secure design patterns, which can help mitigate known security risks.

Mitigation Strategies:

- Threat Modeling: Regularly perform threat modeling to identify and address potential security risks during the design phase.
- Security Design Reviews: Conduct thorough security design reviews to ensure all security requirements are met.
- Use of Secure Design Principles: Implement secure design principles and patterns, such as least privilege, defense in depth, and secure defaults.
- Regular Security Training: Provide regular security training for developers to increase awareness and understanding of secure design practices.
- Insecure Design is a critical aspect of the OWASP Top 10, emphasizing the need for robust security considerations from the very beginning of the application development lifecycle.

Question:



Question 1: Try to reset joseph's password. Keep in mind the method used by the site to validate if you are indeed joseph. **Answer:** No Need to Answer

Question 2: What is the value of the flag in joseph's account? **Answer:** THM{Not_3ven_c4tz_c0uld_sav3_U!}.

5.Security Misconfiguration

Security Misconfiguration is a significant risk highlighted in the OWASP Top 10, often resulting from inadequate security settings or the lack of proper maintenance. It occurs when security controls are not fully implemented or are incorrectly configured, leaving applications vulnerable to attacks. Misconfigurations can happen at any level of an application stack, including network services, platforms, web servers, databases, frameworks, and custom code.

Key Points:

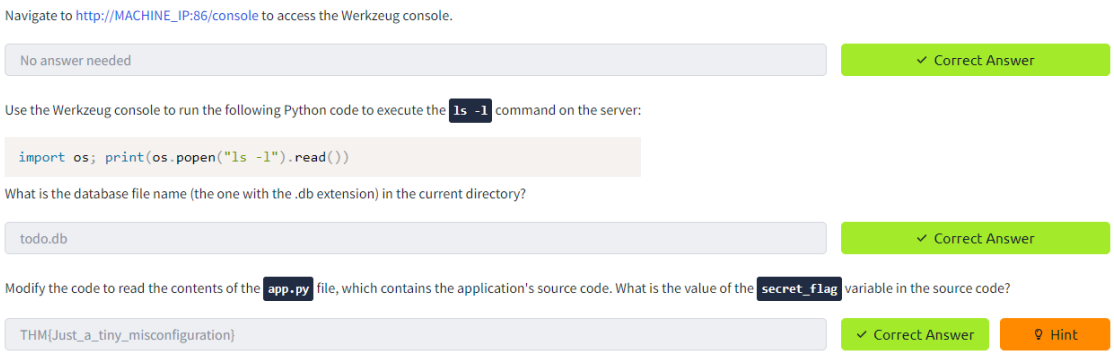
- Default Configurations: Using default settings, which can include weak passwords, unnecessary features, and default accounts that attackers can easily exploit.
- Incomplete Configurations: Failing to properly configure security settings, such as access controls and permission settings.
- Unpatched Systems: Not regularly applying security patches and updates, leaving known vulnerabilities unaddressed.
- Overly Permissive Settings: Granting excessive privileges or permissions that are not necessary for normal operation, increasing the risk of unauthorized access.

Mitigation Strategies:

- Secure Configuration Management: Establish and maintain secure configurations for all components, including operating systems, frameworks, libraries, and applications.
- Regular Audits and Reviews: Conduct regular security audits and reviews to identify and rectify misconfigurations.

*Automated Tools: Utilize automated tools to scan for misconfigurations and enforce security policies.

- Patch Management: Implement a robust patch management process to ensure all components are up-to-date with the latest security patches.
- Principle of Least Privilege: Apply the principle of least privilege, ensuring that users and systems have only the permissions they need to perform their functions.
- Security Misconfiguration underscores the need for diligent and continuous management of security settings across all levels of an application to prevent potential exploitation.



Question :1 Navigate to http://MACHINE_IP:86/console to access the Werkzeug console.
Answer: No Need to Answer

Question 2: Use the Werkzeug console to run the following Python code to execute the ls -l command on the server: **Answer:** todo.db

Question 3: 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:** THM{Just_a_tiny_misconfiguration}

6.Vulnerable and Outdated Components

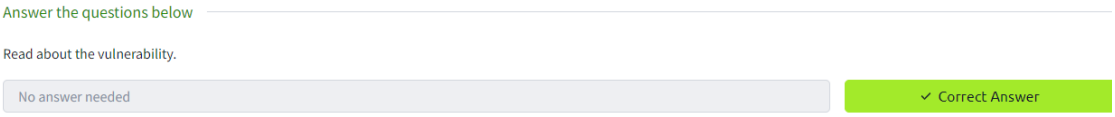
Vulnerable and Outdated Components Vulnerable and Outdated Components refer to using software libraries, frameworks, and other components with known security flaws or those that are no longer maintained. These components can introduce significant security risks to an application, as attackers can exploit the known vulnerabilities to compromise the system.

Key Points:

- Known Vulnerabilities: Components with publicly known vulnerabilities that have not been patched or updated, making them easy targets for attackers.
- Unmaintained Software: Using software that is no longer supported or updated, leading to an accumulation of unaddressed security issues.
- Dependency Management: Failure to track and manage dependencies, resulting in the use of insecure versions of libraries and frameworks.
- Component Testing: Lack of thorough testing for components integrated into the application, increasing the risk of introducing vulnerabilities.

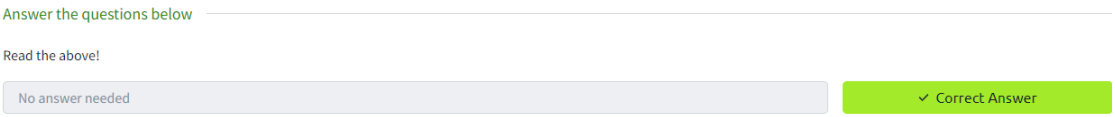
Mitigation Strategies:

- Regular Updates: Keep all components up-to-date with the latest security patches and updates.
- Automated Tools: Utilize automated tools to regularly scan for and identify vulnerable components within your application.
- Dependency Management: Implement robust dependency management practices to ensure that all dependencies are tracked and updated appropriately.
- Thorough Testing: Conduct comprehensive testing, including security testing, for all components before integrating them into the application.
- Vendor Security Policies: Ensure that third-party vendors have strong security policies and practices in place to maintain the security of their components.
- Vulnerable and Outdated Components highlight the importance of maintaining up-to-date and secure components to protect applications from known and emerging threats.

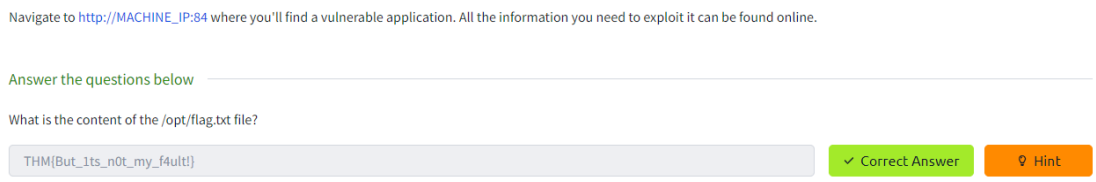


Question 1: Read about the vulnerability. **Answer:** No Need to Answer

Vulnerable and Outdated Components - Exploit



Question 1: What is the content of the /opt/flag.txt file? **Answer:** No Need to Answer



Question 1: What is the content of the /opt/flag.txt file? **Answer:** THM{But_1ts_n0t_my_f4ult!}

7.Identification and Authentication Failures

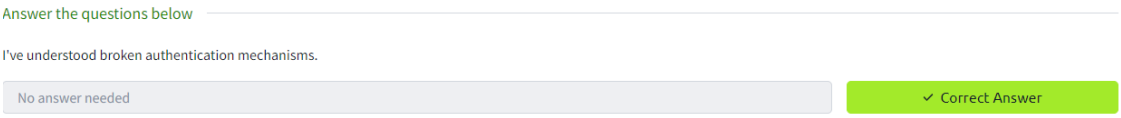
Identification and Authentication Failures occur when an application improperly manages or implements authentication mechanisms. These failures can lead to unauthorized access, allowing attackers to assume the identities of legitimate users. This category highlights the importance of robust and secure authentication methods to safeguard sensitive data and resources.

Key Points:

- Weak Password Policies: Implementing insufficient password policies, such as allowing weak passwords or not enforcing multi-factor authentication.
- Brute Force Attacks: Lack of protection against brute force attacks, where attackers systematically try different combinations to guess passwords.
- Session Management Flaws: Poor session management, which can result in session hijacking or fixation, allowing attackers to gain unauthorized access.
- Improper Credential Storage: Storing credentials in an insecure manner, such as plain text, making it easier for attackers to obtain and use them.

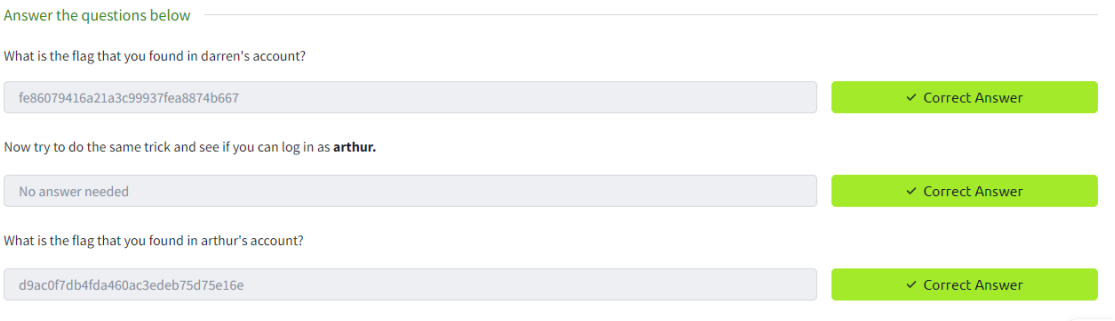
Mitigation Strategies:

- Strong Password Policies: Enforce the use of strong, complex passwords and implement multi-factor authentication to add an extra layer of security.
- Account Lockout Mechanisms: Implement account lockout mechanisms to protect against brute force attacks, locking accounts after a certain number of failed login attempts.
- Secure Session Management: Use secure session management techniques, such as session timeouts and secure cookies, to prevent session hijacking and fixation.
- Secure Credential Storage: Store credentials securely by hashing and salting passwords, and never store them in plain text.
- Identification and Authentication Failures emphasize the need for secure authentication practices to prevent unauthorized access and ensure that only legitimate users can access the application.



Question 1:I've understood broken authentication mechanisms. file? **Answer:** No Need to Answer

Identification and Authentication Failures Practical



Question 1: What is the flag that you found in darren's account? **Answer:** fe86079416a21a3c99937fea8874b667

Question 2: Now try to do the same trick and see if you can log in as arthur. **Answer:** No Need To Answer

Question 3: What is the flag that you found in arthur's account?What is the flag that you found in arthur's account? **Answer:** d9ac0f7db4fda460ac3edeb75d75e16e

8.Software and Data Integrity Failures

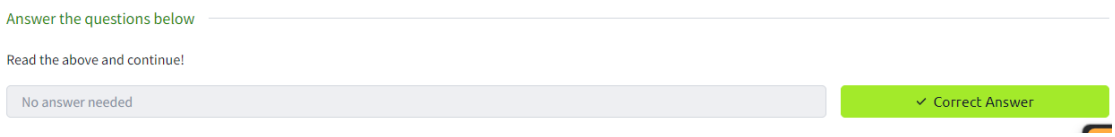
Software and Data Integrity Failures occur when applications fail to protect against integrity violations. These failures can lead to unauthorized manipulation of software and data, compromising the security and functionality of applications. This risk involves the lack of mechanisms to detect, prevent, and respond to unauthorized changes in software and data.

Key Points:

- Untrusted Software Updates: Failure to verify the integrity and authenticity of software updates, allowing attackers to introduce malicious code.
- Code and Data Tampering: Insufficient protection against unauthorized code changes or data manipulation, leading to potential vulnerabilities.
- Lack of Integrity Checks: Absence of integrity checks such as digital signatures and checksums, making it difficult to detect tampering.
- Supply Chain Attacks: Threats that exploit vulnerabilities within the software supply chain, leading to compromised software components.

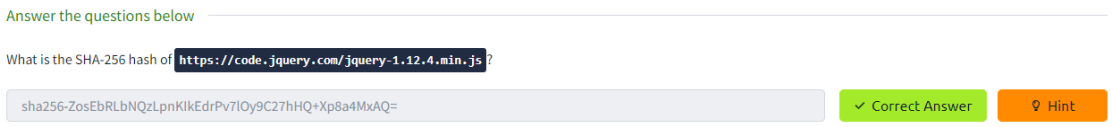
Mitigation Strategies:

- Secure Software Update Mechanisms: Ensure that software updates are delivered securely, using cryptographic signing to verify the integrity and authenticity of updates.
- Implement Integrity Controls: Use integrity controls such as digital signatures, checksums, and hash functions to detect unauthorized modifications to software and data.
- Monitor for Tampering: Regularly monitor software and data for signs of tampering, employing tools and techniques to detect and respond to integrity violations.
- Supply Chain Security: Strengthen supply chain security by vetting third-party components and enforcing security policies for vendors.
- Software and Data Integrity Failures highlight the importance of maintaining the integrity of software and data to prevent unauthorized changes that could compromise the security and reliability of applications.



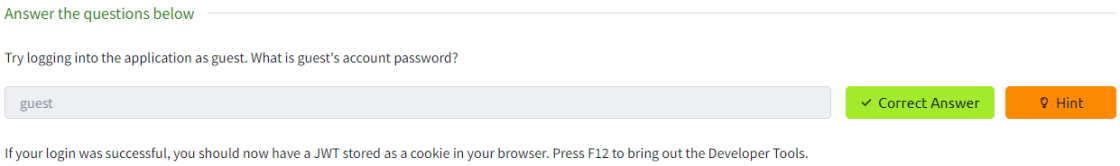
Question 1: Read the above and continue! **Answer:** No Need to Answer

Software Integrity Failures

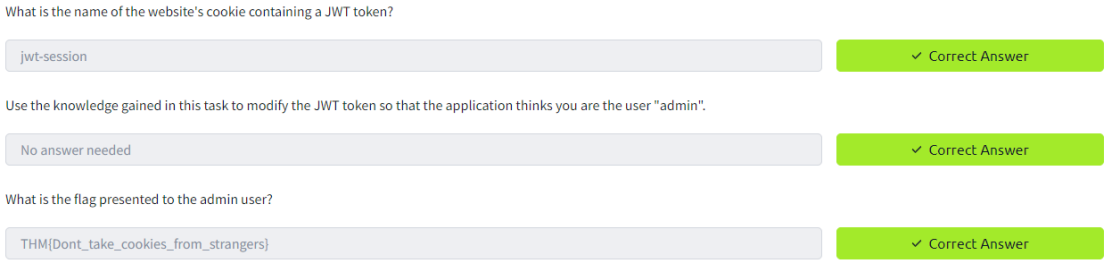


Question 2: What is the SHA-256 hash of (https://code.jquery.com/jquery-1.12.4.min.js?)
Answer: sha256-ZosEbRLbNQzLpnKIkEdrPv7lOy9C27hHQ+Xp8a4MxAQ=

Data Integrity Failures



Question 3: Try logging into the application as guest. What is guest's account password?
Answer: guest



Question 4: What is the name of the website's cookie containing a JWT token? **Answer:** jwt-session

Question 5: Use the knowledge gained in this task to modify the JWT token so that the application thinks you are the user "admin". **Answer:** No Need To Answer

Question 6: What is the flag presented to the admin user? **Answer:** THM{Dont_take_cookies_from_strangers}

9.Security Logging & Monitoring Failures

Security Logging & Monitoring Failures refer to the inadequate implementation of logging and monitoring mechanisms within an application. These failures can hinder the detection of security breaches, compromise response efforts, and leave systems vulnerable to prolonged exploitation. Effective logging and monitoring are crucial for identifying, analyzing, and responding to security incidents in a timely manner.

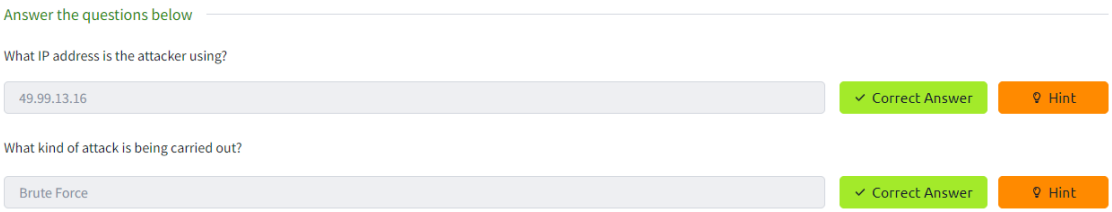
Key Points:

- Insufficient Logging: Lack of detailed logs that capture critical security-related events, such as access attempts, errors, and system changes.

- Delayed Detection: Failure to detect and respond to security incidents promptly, allowing attackers to exploit vulnerabilities for extended periods.
- Inadequate Monitoring: Lack of continuous monitoring of logs and security events, which can delay the identification of suspicious activities.
- Poor Log Management: Inefficient storage, retrieval, and analysis of log data, making it challenging to investigate incidents effectively.

Mitigation Strategies:

- Comprehensive Logging: Implement comprehensive logging that captures detailed information about security events, access attempts, and system changes.
- Real-Time Monitoring: Use real-time monitoring tools to continuously monitor logs and detect suspicious activities as they occur.
- Alerting and Incident Response: Establish alerting mechanisms that notify security teams of potential incidents, and develop a robust incident response plan.
- Log Retention Policies: Define log retention policies to ensure that logs are stored securely and can be retrieved for analysis when needed.
- Regular Audits: Conduct regular audits of logging and monitoring systems to ensure they are functioning correctly and capturing the necessary information.
- Security Logging & Monitoring Failures highlight the importance of implementing robust logging and monitoring practices to detect and respond to security incidents swiftly and effectively.



Question 1: What IP address is the attacker using? **Answer:** 49.99.13.16

Question 2: What kind of attack is being carried out? **Answer:** Brute Force

10.Server-Side Request Forgery (SSRF)

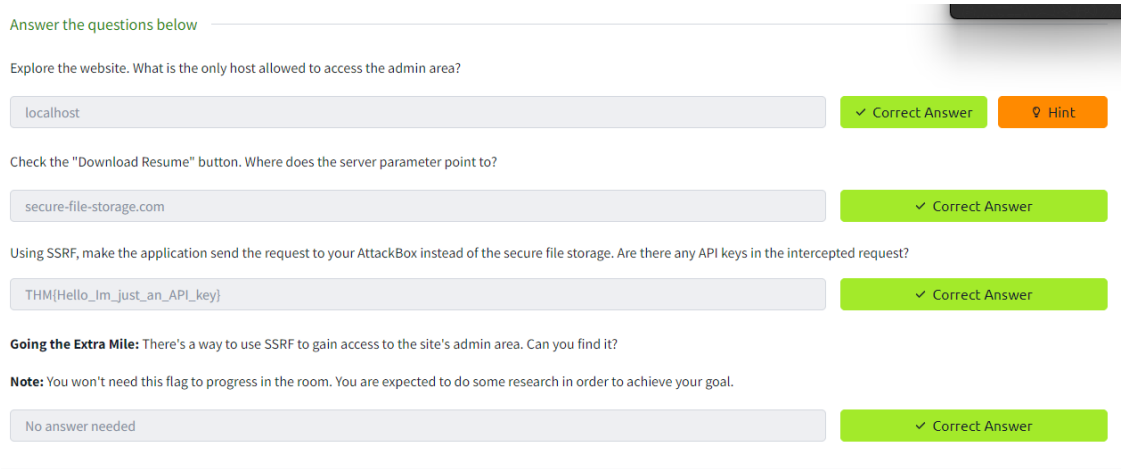
Server-Side Request Forgery (SSRF) occurs when an attacker tricks a server into making a request to an unintended location. This vulnerability arises when a web application is unable to validate or sanitize user input, allowing the attacker to make requests to internal systems, external systems, or the local network.

Key Points:

- Internal Network Exploitation: Attackers can use SSRF to access internal systems that are not exposed to the internet, potentially leading to further exploitation.
- Data Exfiltration: SSRF can be used to extract sensitive data from internal systems or databases by making unauthorized requests.
- Bypassing Firewall Rules: Attackers may use SSRF to bypass firewall rules and access restricted areas of the network.
- Service Interaction: Manipulating the server to interact with unintended services, potentially leading to unauthorized actions or information disclosure.

Mitigation Strategies:

- Input Validation: Implement strict input validation to ensure that user input does not contain malicious URLs or unexpected request targets.
- Allowlisting: Use allowlists to restrict requests to only trusted and necessary destinations, preventing unauthorized access.
- Disable Unnecessary Services: Disable any unnecessary internal services that do not need to be exposed to prevent potential exploitation.
- Network Segmentation: Segment the network to ensure that internal and sensitive systems are isolated and not directly accessible by the web server.
- Monitoring and Logging: Implement robust monitoring and logging mechanisms to detect and respond to suspicious or unauthorized requests.
- Server-Side Request Forgery (SSRF) underscores the importance of validating user input and implementing network security best practices to prevent unauthorized access and protect sensitive systems.



Question 1: Explore the website. What is the only host allowed to access the admin area?
Answer: localhost

Question 2: Check the “Download Resume” button. Where does the server parameter point to? **Answer:** secure-file-storage.com

Question 3: 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? **Answer:** THM{Hello_Im_just_an_API_key}

Question 4: Going the Extra Mile: There’s a way to use SSRF to gain access to the site’s admin area. Can you find it? **note:** You won’t need this flag to progress in the room. You are expected to do some research in order to achieve your goal. **Answer:** No Need To Answer