The SecOps Group CAP Practice Questions

Shared by Berg on 26-10-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

Based on the below request/response, which of the following statements is true?

Send

GET /dashboard.php?purl=http://attacker.com HTTP/1.1

Host: example.com

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) Firefox/107.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8

Accept-Language: en-GB,en;q=0.5

Accept-Encoding: gzip, deflate

Upgrade-Insecure-Requests: 1

Sec-Fetch-Dest: document

Sec-Fetch-Mode: navigate

Sec-Fetch-Site: none

Sec-Fetch-User: ?1

Cookie: JSESSIONID=38RB5ECV10785B53AF29816E92E2E50

Te: trailers

Connection: keep-alive

Pretty Raw | Hex | php | curl | ln | Pretty

HTTP/1.1 302 Found 2022-12-03 17:38:18 GMT

Date: Sat, 03 Dec 2022 17:38:18 GMT

Server: Apache/2.4.54 (Unix) OpenSSL/1.0.2k-fips PHP/8.0.25

X-Powered-By: PHP/8.0.25

Content-Length: 0

Content-Type: text/html; charset=UTF-8

Connection: keep-alive

Location: http://attacker.com

Set-Cookie: JSESSIONID=38C5ECV10785B53AF29816E92E2E50; Path=/; HttpOnly

Options:

A- Application is likely to be vulnerable to Open Redirection vulnerability
B- Application is vulnerable to Cross-Site Request Forgery vulnerability
C- Application uses an insecure protocol
D- All of the above

Answer:

A

Explanation:

The request is a GET to /dashboard.php with a purl parameter (http://attacker.com). The response is a 302 Found redirect with a Location: http://attacker.com header, indicating the server redirects the client to the URL specified in the purl parameter. Let's evaluate the statements:

Option A ('Application is likely to be vulnerable to Open Redirection vulnerability'): Correct. Open Redirection occurs when an application redirects to a user-supplied URL without validation. Here, the purl parameter (http://attacker.com) is directly used in the Location header, allowing an attacker to redirect users to a malicious site (e.g., for phishing). This is a classic Open Redirection vulnerability if the application does not restrict redirects to trusted domains.

Option B ('Application is vulnerable to Cross-Site Request Forgery vulnerability'): Incorrect. CSRF involves tricking a user into making an unintended request (e.g., via a malicious form). This response does not indicate a CSRF issue; there's no evidence of state-changing actions or lack of CSRF tokens.

Option C ('Application uses an insecure protocol'): Incorrect. The request is made over HTTP, and the redirect is to an HTTP URL (http://attacker.com), which is insecure, but the response itself does not indicate the protocol used for the initial request. The server could be using HTTPS for the initial response; the insecure protocol is in the redirect destination, which relates to the Open Redirection issue, not the application's protocol usage broadly.

Option D ('All of the above'): Incorrect, as only A is true.

The correct answer is A, aligning with the CAP syllabus under 'Open Redirection Vulnerabilities' and 'URL Redirection Attacks.'

Question 2

Question Type: MultipleChoice

Which HTTP header is used by the CORS (Cross-origin resource sharing) standard to control access to resources on a server?

Options:

A- Access-Control-Request-Headers
B- The docker-compose.yml file is a YAML file that contains the server logs and user session information including but not limited to admin users.
C- Access-Control-Allow-Headers
D- Access-Control-Request-Method

Answer:

C

Explanation:

Cross-Origin Resource Sharing (CORS) is a security mechanism that allows servers to specify which origins can access their resources, relaxing the Same-Origin Policy (SOP) for legitimate cross-origin requests. CORS uses specific HTTP headers to control this access. The key header for controlling access to resources is Access-Control-Allow-Origin, which specifies which origins are permitted to access the resource. However, among the provided options, the closest related header is Access-Control-Allow-Headers, which is part of the CORS standard and controls which request headers can be used in the actual request (e.g., during a preflight OPTIONS request).

Option A ('Access-Control-Request-Method'): This header is sent by the client in a preflight request to indicate the HTTP method (e.g., GET, POST) that will be used in the actual request. It is not used by the server to control access.

Option B ('Access-Control-Request-Headers'): This header is sent by the client in a preflight request to list the headers it plans to use in the actual request. It is not used by the server to control access.

Option C ('Access-Control-Allow-Headers'): This header is sent by the server in response to a preflight request, specifying which headers are allowed in the actual request. While Access-Control-Allow-Origin is the primary header for controlling access, Access-Control-Allow-Headers is part of the CORS standard to manage header-based access control, making this the best match among the options.

Option D ('None of the above'): Incorrect, as Access-Control-Allow-Headers is a CORS header.

The correct answer is C, aligning with the CAP syllabus under 'CORS Security' and 'HTTP Headers.'

Question 3

Question Type: MultipleChoice

After purchasing an item on an e-commerce website, a user can view his order details by visiting the URL:

https://example.com/order_id=53870

A security researcher pointed out that by manipulating the order_id value in the URL, a user can view arbitrary orders and sensitive information associated with that order_id.

Which of the following is correct?

Options:

A- The root cause of the problem is a lack of input validation and by implementing a strong whitelisting, the problem can be solved
B- The root cause of the problem is a weak authorization (Session Management) and by validating a user's privileges, the issue can be fixed
C- The problem can be solved by implementing a Web Application Firewall (WAF)
D- The browser contains both the public and private key of all known Certifying Authorities (CA) and based on that it is able to differentiate between a valid and an invalid TLS Certificate

Answer:

B

Explanation:

The scenario describes an e-commerce website where a user can view order details by manipulating the order_id parameter in the URL (e.g., https://example.com/order_id=53870). A security researcher found that changing the order_id allows access to arbitrary orders and sensitive data, indicating an authorization issue. This is not a simple input validation problem (e.g., SQL injection or path traversal), as the input (order_id) is processed, but the system fails to enforce proper access controls. This is a classic case of Insecure Direct Object Reference (IDOR), where an attacker can access resources by guessing or manipulating identifiers without proper authorization checks. The root cause is a weak authorization mechanism, likely tied to poor session management or privilege validation, allowing unauthorized users to view others' data.

Option A ('The root cause of the problem is a lack of input validation...'): Incorrect, as the issue is not about invalid input (e.g., malicious code) but about unauthorized access to valid data. Whitelisting might help sanitize input, but it doesn't address authorization.

Option B ('The root cause of the problem is a weak authorization (Session Management)...'): Correct, as the problem stems from inadequate authorization checks. Validating user privileges (e.g., ensuring the user can only access their own orders) or improving session management (e.g., tying orders to user sessions) can fix this IDOR vulnerability.

Option C ('The problem can be solved by implementing a Web Application Firewall (WAF)'): Incorrect as a standalone solution, as WAFs mitigate certain attacks (e.g., SQLi, XSS) but are not a substitute for fixing authorization logic. A WAF might detect patterns but won't enforce proper access control.

Option D ('None of the above'): Incorrect, as Option B accurately identifies the root cause and solution.

The correct answer is B, aligning with the CAP syllabus under 'Authorization and Access Control' and 'OWASP Top 10 (A04:2021 - Insecure Design).'

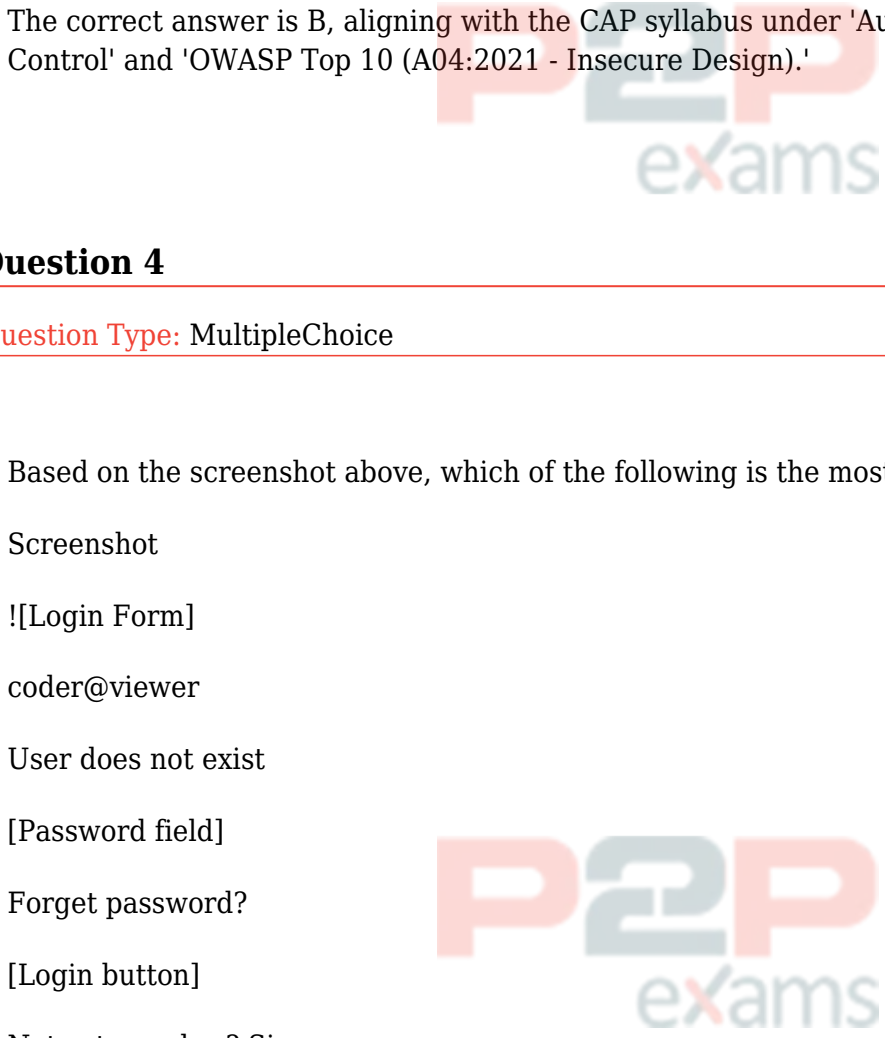
P2P exams

Question 4

Question Type: MultipleChoice

Based on the screenshot above, which of the following is the most true?

Screenshot



P2P exams

Options:

A- The application is vulnerable to username enumeration
B- The application is vulnerable to brute-force attacks
C- The change password feature does not validate the user
D- The application does not enforce a strong password policy

Answer:

A

Explanation:

The screenshot shows a login form where the user coder@viewer attempts to log in, and the application responds with 'User does not exist.' Let's evaluate the statements:

Option A ('The application is vulnerable to username enumeration'): Correct. Username enumeration occurs when an application reveals whether a username exists in the system, often through distinct error messages. Here, the message 'User does not exist' for coder@viewer directly indicates that the username is invalid, allowing an attacker to enumerate valid usernames by testing different inputs and observing the responses (e.g., 'Invalid password' for existing users vs. 'User does not exist'). Best practice is to use generic error messages like 'Invalid username or password' to prevent enumeration.

Option B ('The application is vulnerable to brute-force attacks'): Incorrect. There's no evidence in the screenshot of a lack of brute-force protections (e.g., rate limiting, account lockout). Brute-force vulnerability would require additional context, such as no CAPTCHA or no lockout mechanism, which is not shown.

Option C ('The application does not enforce a strong password policy'): Incorrect. The screenshot does not provide information about password requirements (e.g., length, complexity), so we cannot conclude whether a strong password policy is enforced.

Option D ('None of the above'): Incorrect, as A is true.

The correct answer is A, aligning with the CAP syllabus under 'Username Enumeration' and 'Authentication Security.'

Question 5

Question Type: MultipleChoice

Multifactor authentication will NOT be able to prevent:

Options:

A- Cross-Site Scripting Vulnerability
B- Cross-Site Request Forgery Vulnerability
C- Path Traversal Vulnerability
D- CORS is exploitable if the value of the HTTP headers are Access-Control-Allow-Origin: * and Access-Control-Allow-Credentials: true

Answer:

D

Explanation:

Multifactor Authentication (MFA) enhances security by requiring multiple forms of verification (e.g., something you know, like a password, and something you have, like a one-time code) to authenticate a user. It is effective against attacks that rely on stolen credentials, but let's evaluate its impact on the listed vulnerabilities:

Option A ('Cross-Site Scripting Vulnerability'): XSS (Cross-Site Scripting) involves injecting malicious scripts into a web application that execute in the victim's browser. MFA does not prevent XSS because it occurs after authentication; an attacker can exploit XSS to steal session cookies or perform actions on behalf of the authenticated user, bypassing MFA's protection.

Option B ('Cross-Site Request Forgery Vulnerability'): CSRF (Cross-Site Request Forgery) tricks a user's browser into making unintended requests to a site where they are authenticated. MFA does not prevent CSRF because the attack leverages the user's existing session (post-authentication). The browser automatically sends cookies (e.g., session cookies) with the forged request, and MFA does not interfere with this process.

Option C ('Path Traversal Vulnerability'): Path Traversal allows an attacker to manipulate file paths (e.g., ../../etc/passwd) to access unauthorized files on the server. MFA does not prevent this because it is an application-level vulnerability unrelated to user authentication; an attacker with or without credentials can exploit it if the application fails to validate input.

Option D ('All of the above'): Correct, as MFA is designed to secure the authentication process, not to mitigate vulnerabilities like XSS, CSRF, or Path Traversal, which exploit application logic or session management after authentication.

The correct answer is D, aligning with the CAP syllabus under 'Multifactor Authentication' and 'Application Security Vulnerabilities.'

Question 6

Question Type: MultipleChoice

What is the full form of SAML?

Options:

A- Security Assertion Markup Language
B- Security Authorization Markup Language

C- Security Assertion Management Language
D- Secure Authentication Markup Language

Answer:

A

Explanation:

SAML (Security Assertion Markup Language) is an open standard for exchanging authentication and authorization data between parties, particularly in the context of single sign-on (SSO). It is based on XML and is widely used to enable secure web-based authentication and authorization across different domains. The correct full form is Security Assertion Markup Language, where 'Assertion' refers to statements about a subject (e.g., identity, attributes), 'Markup' indicates the XML-based structure, and 'Language' denotes the defined syntax.

Option A ('Security Assertion Markup Language'): This is the correct and official full form of SAML as defined by OASIS (Organization for the Advancement of Structured Information Standards).

Option B ('Security Authorization Markup Language'): Incorrect, as 'Authorization' is not part of the acronym; SAML focuses on both authentication and authorization assertions.

Option C ('Security Assertion Management Language'): Incorrect, as 'Management' is not part of the acronym; SAML is about markup, not management.

Option D ('Secure Authentication Markup Language'): Incorrect, as 'Secure' is not part of the acronym, and SAML covers more than just authentication.

The correct answer is A, aligning with the CAP syllabus under 'Authentication and Authorization' and 'Single Sign-On (SSO) Standards.'

Question 7

Question Type: MultipleChoice

The application is vulnerable to Cross-Site Scripting. Which of the following exploitation is NOT possible at all?

Options:

A- Steal the user's session identifier stored on a non HttpOnly cookie
B- Steal the contents from the web page

C- Steal the contents from the application's database
D- Steal the contents from the user's keystrokes using keyloggers

Answer:

C

Explanation:

Cross-Site Scripting (XSS) is a vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. These scripts execute in the context of the victim's browser, enabling various exploitations. Let's evaluate each option:

Option A ('Steal the user's session identifier stored on a non HttpOnly cookie'): This is possible with XSS. If a session cookie is not marked as HttpOnly (preventing JavaScript access), an attacker can use a script to access document.cookie and steal the session ID, leading to session hijacking.

Option B ('Steal the contents from the web page'): This is also possible. An XSS payload can manipulate the DOM, extract content (e.g., via innerHTML), and send it to the attacker, such as through a GET request to a malicious server.

Option C ('Steal the contents from the application's database'): This is not possible with XSS alone. XSS operates on the client side within the browser's sandbox and cannot directly access the server-side database. Database access requires server-side vulnerabilities (e.g., SQL injection), which is a separate attack vector. Thus, this exploitation is not feasible through XSS.

Option D ('Steal the contents from the user's keystrokes using keyloggers'): This is possible. An XSS script can inject a keylogger (e.g., using onkeydown events) to capture keystrokes and transmit them to the attacker, especially on pages where sensitive data (e.g., forms) is entered.

Therefore, the correct answer is C, as XSS cannot directly exploit the database. This distinction is crucial in understanding attack vectors, a core topic in the CAP syllabus under 'OWASP Top 10 (A03:2021 - Injection)' and 'XSS Mitigation.'

Question 8

Question Type: MultipleChoice

Given the Race Condition vulnerability, which of the following statements is true?

Options:

A- A situation that occurs when two threads access the same resource at the same time.
B- A situation that occurs when two threads access different resources at the same time.
C- A situation that occurs when a single thread unpredictably accesses two resources.
D- A situation that occurs when a single thread predictably accesses two resources.

Answer:

A

Explanation:

A Race Condition vulnerability occurs in multi-threaded or multi-process applications when two or more threads access a shared resource concurrently, and the outcome depends on the non-deterministic order of their execution. This can lead to inconsistent states or security issues, such as privilege escalation or data corruption, if the access is not properly synchronized (e.g., using locks or semaphores). The classic definition focuses on concurrent access to the same resource.

Option A ('A situation that occurs when two threads access the same resource at the same time'): Correct, as this accurately describes a race condition where the lack of synchronization on a shared resource (e.g., a file, variable, or database entry) can lead to unpredictable behavior.

Option B ('A situation that occurs when two threads access different resources at the same time'): Incorrect, as race conditions specifically involve contention over the same resource, not different ones.

Option C ('A situation that occurs when a single thread unpredictably accesses two resources'): Incorrect, as race conditions require multiple threads or processes; a single thread's behavior is not a race condition.

Option D ('A situation that occurs when a single thread predictably accesses two resources'): Incorrect, as predictability negates the race condition concept, and it still involves only one thread.

The correct answer is A, aligning with the CAP syllabus under 'Race Condition Vulnerabilities' and 'Multi-Threaded Security.'

Question 9

Question Type: MultipleChoice

Which option best attributes is NOT used to secure the cookie?

Options:

A- HttpOnly
B- Secure
C- Restrict
D- Same-Site

Answer:

C

Explanation:

Cookies can have security attributes to protect them against various attacks. Let's evaluate each option to determine which attribute is not used to secure cookies:

Option A ('HttpOnly'): The HttpOnly attribute prevents cookies from being accessed by JavaScript (e.g., via document.cookie). This mitigates XSS attacks that attempt to steal session cookies, making it a valid security attribute.

Option B ('Secure'): The Secure attribute ensures that the cookie is only sent over HTTPS connections, preventing it from being transmitted over unencrypted HTTP. This protects against interception (e.g., in a man-in-the-middle attack), making it a valid security attribute.

Option C ('Restrict'): There is no standard cookie attribute called Restrict. Cookie security attributes are well-defined (e.g., HttpOnly, Secure, SameSite), and Restrict does not exist in this context. This is not a valid attribute for securing cookies.

Option D ('Same-Site'): The SameSite attribute (e.g., SameSite=Strict or SameSite=Lax) controls whether a cookie is sent with cross-site requests. It helps mitigate CSRF attacks by ensuring the cookie is only sent with same-site requests (or limited cross-site scenarios), making it a valid security attribute.

The correct answer is C, as Restrict is not a recognized cookie attribute, aligning with the CAP syllabus under 'Cookie Security' and 'Session Management.'

To Get Premium Files for CAP Visit

https://www.p2pexams.com/products/cap

For More Free Questions Visit

https://www.p2pexams.com/the-secops-group/pdf/cap

