ISC.CAP.v2025-08-19.q27

Exam Code:	CAP
Exam Name:	Certified AppSec Practitioner Exam
Certification Provider:	ISC
Free Question Number:	27
Version:	v2025-08-19
# of views:	102
# of Questions views:	271
https://www.freecram.net/torrent/ISC.CAP.v2025-08-19.q27.html	

NEW QUESTION: 1
Based on the below HTTP request, which of the following statements is correct?
POST /changepassword HTTP/2
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:107.0) Gecko/20100101 Firefox/107.0 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: same-origin Cookie: JSESSIONID=38RB5ECV10785B53AF29816E92E2E50 Content-Length: 95
new_password=usher!@22&confirm_password=usher!@22
A. The change password feature does not validate the user
B. The change password feature uses basic authorization
C. The change password feature is vulnerable to Cross-Site Request Forgery attack
D. All of the above
Answer: (SHOW ANSWER)
The HTTP request is a POST to /changepassword with a session cookie (JSESSIONID) and parameters new_password and confirm_password. Let's evaluate each option:
* Option A ("The change password feature does not validate the user"): The request includes a JSESSIONID cookie, which typically indicates that the user is authenticated via a session. There's no evidence that user validation is absent, so this is not correct.
* Option B ("The change password feature uses basic authorization"): Basic authorization would involve an Authorization: Basic header with a Base64-encoded username and password, which is not present here. The authentication appears to be session-based (via cookie), not basic auth, so this is incorrect.
* Option C ("The change password feature is vulnerable to Cross-Site Request Forgery attack"): Cross-Site Request Forgery (CSRF) occurs when a malicious site tricks a user's browser into making an unintended request to another site where the user is authenticated. This request lacks a CSRF token (e.

g., a unique, unpredictable token in the request body or header) to verify the request's legitimacy. The Sec-Fetch-Site: same-origin header indicates the request is currently from the same origin, but this is a browser feature, not a server-side CSRF protection. Without a CSRF token, the endpoint is vulnerable to CSRF, as an attacker could craft a malicious form on another site to submit this request on behalf of the user. This is the correct answer.
* Option D ("All of the above"): Since A and B are incorrect, D cannot be correct.
The correct answer is C, aligning with the CAP syllabus under "Cross-Site Request Forgery (CSRF)" and
"OWASP Top 10 (A08:2021 - Software and Data Integrity Failures)."References: SecOps Group CAP Documents - "CSRF Prevention," "Session Management," and "OWASP Secure Coding Practices" sections.

NEW QUESTION: 2
The following request is vulnerable to Cross-Site Request Forgery vulnerability.
POST /changepassword HTTP/2Host: example.com User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) rv:107.0) Gecko/20100101 Firefox/107.0 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec- Fetch-Site: same-origin Cookie: JSESSIONID=38RC5ECV10785B53AF19816E92E2E50 Content-Length: 95 new_password=lov3MyPiano23&confirm_password=lov3MyPiano23
A. True
B. False
Answer: (SHOW ANSWER)
Cross-Site Request Forgery (CSRF) occurs when an attacker tricks a user's browser into making an unintended request to a site where the user is authenticated, potentially performing actions like changing a password. Let's analyze the request:
* The request is a POST to /changepassword with a Cookie: JSESSIONID, indicating the user is authenticated via a session. The Content-Length: 95 and payload (new_password=lov3MyPiano23&confirm_password=lov3MyPiano23) suggest a state-changing operation (password change).
* CSRF vulnerability arises when the request lacks a unique, unpredictable token to validate its legitimacy, and the server accepts it based solely on the session cookie. The request includes no CSRF token (e.g., in the body or headers like X-CSRF-Token).
* The Sec-Fetch-Site: same-origin header indicates the request originates from the samedomain, but this is a browser feature and does not guarantee server-side protection against CSRF from a malicious site (e.
g., via a hidden iframe or form submission).
* Without a CSRF token, an attacker could craft a malicious HTML page with a form that submits this exact request when a victim visits their site while authenticated to example.com, exploiting the browser' s automatic inclusion of the JSESSIONID cookie. This is a textbook CSRF vulnerability.

* Option A ("True"): Correct, as the request lacks a CSRF token, making it vulnerable to CSRF attacks.
* Option B ("False"): Incorrect, as the absence of a CSRF token indicates vulnerability.
The correct answer is A, aligning with the CAP syllabus under "Cross-Site Request Forgery (CSRF)" and
"Session Management."References: SecOps Group CAP Documents - "CSRF Prevention," "Session Security," and "OWASP CSRF Prevention Cheat Sheet" sections.

NEW QUESTION: 3
Which HTTP header is used by the CORS (Cross-origin resource sharing) standard to control access to resources on a server?
A. Access-Control-Request-Method
B. Access-Control-Request-Headers
C. Access-Control-Allow-Headers
D. None of the above
Answer: (SHOW ANSWER)
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 isAccess- Control-Allow-Origin, which specifies which origins are permitted to access the resource. However, among the provided options, the closest related header isAccess-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 apreflight 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." References: SecOps Group CAP Documents - "CORS Configuration," "Security Headers," and "OWASP Secure Headers Guide" sections.

NEW QUESTION: 4
Which of the following is considered as a safe password?

A. Monday@123
B. abcdef
C. Sq0Jh819%ak
D. 1234567890
Answer: (SHOW ANSWER)
A safe password must adhere to security best practices, including sufficient length, complexity, and resistance to common attacks (e.g., brute force, dictionary attacks). Let's evaluate each option:
* Option A ("Monday@123"): This password is weak because it combines a common word ("Monday") with a simple number and symbol pattern. It is vulnerable to dictionary attacks and does not meet complexity requirements (e.g., mixed case, special characters, and randomness).
* Option B ("abcdef"): This is a sequence of letters with no numbers, special characters, or uppercase letters. It is extremely weak and easily guessable, making it unsafe.
* Option C ("Sq0Jh819%ak"): This password is considered safe because it is at least 10 characters long, includes a mix of uppercase letters (S, J, H), lowercase letters (q, h, a, k), numbers (0, 8, 9, 1), and a special character (%). It lacks predictable patterns and meets modern password policy standards (e.g., NIST SP 800-63B recommends at least 8 characters with complexity).
* Option D ("1234567890"): This is a simple numeric sequence, highly predictable, and vulnerable to brute-force attacks, making it unsafe.
The correct answer is C, as it aligns with secure password creation guidelines, a key topic in the CAP syllabus under "Authentication Security" and "Secure Coding Practices."References:
SecOps Group CAP Documents
- "Password Management," "Authentication Security," and "OWASP Secure Coding Guidelines" sections.

NEW QUESTION: 5
Based on the screenshot below, which of the following statements is true?
Request
GET /userProfile.php?sessionId=7576572ce164646de967c759643d53031 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=7576572ce164646de967c759643d53031 Te: trailers Connection: keep-alive PrettyRaw | Hex | php | curl | ln | Pretty HTTP/1.1 200 OK Date: Fri, 09 Dec 2022 11:42:27 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: 12746 Content-Type: text/html; charset=UTF-8 Connection: keep-alive Set-Cookie: JSESSIONID=7576572ce164646de967c759643d53031; Path=/; HttpOnly
<html>
<head>

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Example Domain</title>
</head>
<body style="background-color:#f0f0f2; margin:0; padding:0; font-family: -apple-system, system-ui, BlinkMacSystemFont, 'Segoe UI', 'Open Sans', 'Helvetica Neue', Helvetica, Arial, sans-serif;">
...
</body>
</html>

A. The application uses an insecure channel (non-TLS)
B. The application uses an insecure HTTP method (GET) to send sensitive information
C. The application is vulnerable to Cross-Site Scripting attacks
D. All of the above
Answer: (SHOW ANSWER)

The request is a GET to /userProfile.php with a sessionId parameter matching the JSESSIONID cookie, and the response is a 200 OK with an HTML page. Let's evaluate the statements:
* Option A ("The application uses an insecure channel (non-TLS)"):
The request uses http:// (inferred from the absence of https:// in the Host and GET line), indicating a non-TLS channel. However, the question asks about the "application" (likely the server-side behavior), and the response does not explicitly confirm the channel used for the response. Modern browsers might enforce TLS, but the request suggests an insecure channel. This could be true, but it depends on the server's configuration, making it less certain without further context.
* Option B ("The application uses an insecure HTTP method (GET) to send sensitive information")
: Correct. The sessionId parameter in the URL (/userProfile.php?sessionId=7576572ce164646de967c759643d53031) is sensitive data, as it could be used to hijack the user's session. Using GET exposes this data in browser history, server logs, and referral headers, which is insecure. Best practice is to use POST for sensitive data to avoid such exposure. The JSESSIONID cookie is marked HttpOnly, mitigating some risks, but the sessionId in the URL remains a vulnerability.
* Option C ("The application is vulnerable to Cross-Site Scripting attacks"): The response includes an HTML page with no visible user input or script execution. There's no evidence of unsanitized input or script injection (e.g., no dynamic content like <script> tags with user data). The X-Xss-Protection: 1; mode=block header (though not shown) would mitigate XSS if present, but the response alone does not indicate vulnerability, so this is incorrect.
* Option D ("All of the above"): Incorrect, as only B is definitively true; A is uncertain without confirming the response channel, and C lacks evidence.
The correct answer is B, aligning with the CAP syllabus under "HTTP Methods Security" and "Session Management."References: SecOps Group CAP Documents - "Insecure HTTP Methods," "Sensitive Data Exposure," and "OWASP Secure Coding Practices" sections.

NEW QUESTION: 6
Which of the following Google Dorks can be used for finding directory listing on victim-app.com?
A. intitle:"Index of" site:victim-app.com
B. intext:"Index of" site:victim-app.com
C. Both A and B
D. None of the above
Answer: (SHOW ANSWER)
Google Dorks are advanced search operators used to find specific information or vulnerabilities on the web.
Directory listing vulnerabilities occur when a web server exposes the contents of a directory (e.g., file names, paths) due to misconfiguration. The operators intitle: and intext: are used to search for specific terms in the title or body of web pages, respectively, combined with site: to limit the search to a specific domain.
* Option A ("intitle:'Index of' site:victim-app.com"): Correct, as intitle:"Index of" targets pages with "Index of" in the title, a common indicator of directory listings, and site:victim-app.com restricts the search to that domain.
* Option B ("intext:'Index of' site:victim-app.com"): Correct, as intext:"Index of" searches for "Index of" within the page content, another reliable indicator of directory listings, combined with the domain restriction.
* Option C ("Both A and B"): Correct, as both intitle: and intext: can effectively identify directory listings, making this the most comprehensive answer.
* Option D ("None of the above"): Incorrect, as both A and B are valid Google Dorks for this purpose.
The correct answer is C, aligning with the CAP syllabus under "Reconnaissance Techniques" and "Google Dorking."References: SecOps Group CAP Documents - "Information Gathering," "Google Hacking," and
"OWASP Testing Guide" sections.

NEW QUESTION: 7
Which of the following is NOT a symmetric key encryption algorithm?
A. RC4
B. AES
C. DES
D. RSA
Answer: (SHOW ANSWER)
Symmetric key encryption algorithms use the same key for both encryption and decryption, while asymmetric algorithms use a pair of keys (public and private). Let's evaluate the options:
* Option A ("RC4"): RC4 is a symmetric key encryption algorithm. It is a stream cipher that uses a single key to both encrypt and decrypt data, though it is considered insecure due to known cryptographic weaknesses (e.g., biases in the keystream).

* Option B ("AES"): AES (Advanced Encryption Standard) is a symmetric key encryption algorithm. It uses a single key (e.g., 128, 192, or 256 bits) for both encryption and decryption, widely regarded as secure when properly implemented.
* Option C ("DES"): DES (Data Encryption Standard) is a symmetric key encryption algorithm. It uses a 56-bit key for both encryption and decryption, but it is now considered insecure due to its small key size and vulnerability to brute-force attacks.
* Option D ("RSA"): RSA (Rivest-Shamir-Adleman) is an asymmetric key encryption algorithm. It uses a public key to encrypt and a private key to decrypt, making it an asymmetric algorithm, not a symmetric one.
The correct answer is D, as RSA is the only asymmetric algorithm listed, aligning with the CAP syllabus under "Cryptography Fundamentals" and "Symmetric vs. Asymmetric Encryption."References: SecOps Group CAP Documents - "Symmetric Encryption Algorithms," "Asymmetric Encryption," and "OWASP Cryptographic Storage Cheat Sheet" sections.

NEW QUESTION: 8
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 PrettyRaw | 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
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: (SHOW ANSWER)
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."References: SecOps Group CAP Documents - "Open Redirection," "Input Validation for Redirects," and "OWASP Top 10 (A10:2021 - Server-Side Request Forgery)" sections.

NEW QUESTION: 9
In the screenshot below, which of the following is incorrect?
Target: https://example.com
HTTP/1.1 404 Not Found
Date: Fri, 09 Dec 2022 18:03:49 GMT
Server: Apache
Vary: Cookie
X-Powered-By: PHP/5.4.5-5
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Content-Length: 0
Content-Type: text/html; charset=UTF-8
Cookie: JSESSIONID=1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789; secure; HttpOnly; SameSite=None
A. The application discloses the framework name and version
B. The application reveals user-agent details
C. A cookie is set with HttpOnly and a Secure flag
D. The application accepts insecure protocol
Answer: (SHOW ANSWER)
The screenshot shows an HTTP response header from https://example.com with a 404 status. Let's evaluate each option:

* Option A ("The application discloses the framework name and version"): The X-Powered-By: PHP
/5.4.5-5 header reveals the server is running PHP version 5.4.5-5, which is a security risk as it exposes the framework and version. This information can help attackers identify known vulnerabilities, making A incorrect (i.e., it is a problem).
* Option B ("The application reveals user-agent details"): The response does not include user-agent details; it only shows the server's configuration. User-agent details are part of the request, not the response, so this is incorrect (not a problem here).
* Option C ("A cookie is set with HttpOnly and a Secure flag"): The Cookie header includes HttpOnly and Secure attributes, which are best practices to prevent JavaScript access and ensure transmission over HTTPS, respectively. This is correct behavior, so it is not incorrect.
* Option D ("The application accepts insecure protocol"): The response uses https://, indicating a secure protocol (TLS), and there's no evidence of accepting insecure protocols like HTTP. This is not incorrect.
Thus, the incorrect statement is A, as disclosing the framework name and version via X-Powered-By is a security misconfiguration. This aligns with the CAP syllabus under "Security Headers" and "Information Disclosure."References: SecOps Group CAP Documents - "Security Misconfigurations," "HTTP Headers," and "OWASP Top 10 (A05:2021 - Security Misconfiguration)" sections.

NEW QUESTION: 10

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?

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. None of the above
Answer: (SHOW ANSWER)
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 ofInsecure Direct Object References (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 orprivilege 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)."References: SecOps Group CAP Documents - "Session Management," "Insecure Direct Object References (IDOR)," and "OWASP Top 10" sections.

NEW QUESTION: 11
Determine the primary defense against a SQL injection vulnerability
A. Using a Web Application Firewall (WAF)
B. Prepared Statements with Parameterized Queries
C. Use of NoSQL Database
D. Blacklisting Single Quote Character (')
Answer: (SHOW ANSWER)
SQL Injection (SQLi) occurs when an attacker injects malicious SQL code into a query by manipulating user input (e.g., ' OR '1'='1'), allowing unauthorized data access or manipulation.
Let's evaluate the defenses:
* Option A ("Using a Web Application Firewall (WAF)"): A WAF can detect and block SQL injection attempts by filtering malicious patterns (e.g., ' OR '1'='1'), but it is not the primary defense.
WAFs can be bypassed with sophisticated attacks (e.g., encoded payloads), and they are a secondary layer, not a fix for the root cause in the application code.
* Option B ("Prepared Statements with Parameterized Queries"): Correct. Prepared statements with parameterized queries separate SQL code from user input by using placeholders (e.g., ? in SELECT * FROM users WHERE username = ?). The database engine handles the input as data, not executable code, preventing SQL injection. This is the industry-standard primary defense

(recommended by OWASP and NIST) because it addresses the root cause by ensuring user input cannot alter the query structure.
* Option C ("Use of NoSQL Database"): Switching to a NoSQL database (e.g., MongoDB) does not inherently prevent injection vulnerabilities. NoSQL databases can still be vulnerable to injection (e.g., MongoDB's $where operator), and SQL injection applies to relational databases. This is not a defense against SQLi.
* Option D ("Blacklisting Single Quote Character (')"): Blacklisting specific characters (e.g., ') attempts to block known malicious input, but it is ineffective as a primary defense. Attackers can bypass blacklists using alternate encodings (e.g., %27 for '), comments (e.g., --), or other techniques.
Blacklisting is reactive and prone to evasion, unlike prepared statements.
The correct answer is B, aligning with the CAP syllabus under "SQL Injection Prevention" and "OWASP Top
10 (A03:2021 - Injection)."References: SecOps Group CAP Documents - "SQL Injection Defense," "Secure Coding Practices," and "OWASP SQL Injection Prevention Cheat Sheet" sections.

NEW QUESTION: 12
Which of the following hashing algorithms is considered to be the most secure amongst these?
A. SHA-0
B. MD5
C. SHA-1
D. Bcrypt
Answer: (SHOW ANSWER)
Hashing algorithms are used to securely store passwords by transforming them into fixed-length strings. A secure hashing algorithm for passwords should be resistant to collision attacks, preimage attacks, and brute- force attempts, and should be slow to compute to deter attackers.
Let's evaluate the options:
* Option A ("SHA-0"): SHA-0 is the original version of the SHA family, published in 1993, but it was quickly withdrawn due to serious cryptographic weaknesses (e.g., collision vulnerabilities). It is not secure and should not be used.
* Option B ("MD5"): MD5 is a widely used hash function but is cryptographically broken. It is vulnerable to collision attacks (e.g., practical attacks demonstrated since 2004) and is extremely fast, making it unsuitable for password hashing as it can be brute-forced easily.
* Option C ("SHA-1"): SHA-1, part of the SHA family, is also considered broken for security purposes.
It has known collision vulnerabilities (e.g., the SHAttered attack in 2017 demonstrated practical collisions), and like MD5, it is too fast for secure password hashing.
* Option D ("Bcrypt"): Bcrypt is specifically designed for password hashing. It is a slow hashing algorithm with a configurable work factor (cost factor), making it resistant to brute-force attacks. It also includes a built-in salt to prevent rainbow table attacks. Bcrypt is widely recommended by

security standards (e.g., OWASP, NIST) for secure password storage and is the most secure option among those listed.
The correct answer is D, aligning with the CAP syllabus under "Password Hashing" and "Cryptographic Best Practices."References: SecOps Group CAP Documents - "Secure Password Storage," "Hashing Algorithms," and "OWASP Password Storage Cheat Sheet" sections.

NEW QUESTION: 13
After purchasing an item on an e-commerce website, a user can view their 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. This attack is known as:
A. Insecure Direct Object Reference
B. Session Poisoning
C. Session Riding OR Cross-Site Request Forgery
D. Server-Side Request Forgery
Answer: (SHOW ANSWER)
The scenario describes a vulnerability where a user can manipulate the order_id parameter in theURL (e.g.,
https://example.com/?order_id=53870) to access other users' order details, indicating a lack of proper access control. This is a classic case of anInsecure Direct Object Reference (IDOR)attack. IDOR occurs when an application exposes a reference to an internal object (e.g., an order ID) that can be manipulated by an unauthorized user to access resources they should not have access to, without validating the user's permissions.
* Option A ("Insecure Direct Object Reference"): Correct, as the ability to change order_id to view arbitrary orders fits the definition of IDOR.
* Option B ("Session Poisoning"): Incorrect, as session poisoning involves corrupting or altering a user' s session data, which is not indicated here.
* Option C ("Session Riding OR Cross-Site Request Forgery"): Incorrect, as CSRF involves tricking a user into submitting a request (e.g., via a malicious form), not manipulating a URL parameter directly.
* Option D ("Server-Side Request Forgery"): Incorrect, as SSRF involves tricking the server into making unauthorized requests to internal or external resources, which is not the case here.
The correct answer is A, aligning with the CAP syllabus under "Insecure Direct Object References (IDOR)" and "OWASP Top 10 (A04:2021 - Insecure Design)."References: SecOps Group CAP Documents - "IDOR Vulnerabilities," "Access Control," and "OWASP Testing Guide" sections.

NEW QUESTION: 14

A website administrator forgot to renew the TLS certificate on time and as a result, the application is now displaying a TLS error message. However, on closer inspection, it appears that the error is due to the TLS certificate expiry.
In the scenario described above, which of the following is correct?
A. There is no urgency to renew the certificate as the communication is still over TLS
B. There is an urgency to renew the certificate as the users of the website may get conditioned to ignore TLS warnings and therefore ignore a legitimate warning which could be a real Man-in-the-Middle attack
Answer: (SHOW ANSWER)
This question is identical to Question 52, describing a scenario where a TLS certificate has expired, causing a TLS error message, and asking about the correct course of action. The analysis remains the same:
* Option A ("There is no urgency to renew the certificate as the communication is still over TLS"): Incorrect. An expired TLS certificate invalidates the trust model, even if the connection technically uses TLS. Browsers will issue warnings, and users may bypass them, but the lack of a valid certificate compromises security, making renewal urgent.
* Option B ("There is an urgency to renew the certificate as the users of the website may get conditioned to ignore TLS warnings and therefore ignore a legitimate warning which could be a real Man-in-the-Middle attack"): Correct. Repeated exposure to TLS warnings due to an expired certificate may desensitize users, increasing the risk that they ignore legitimate warnings from a Man-in- the-Middle (MitM) attack. Renewing the certificate promptly is essential to maintain security and user trust.
The correct answer is B, aligning with the CAP syllabus under "TLS Configuration" and "Certificate Management."References: SecOps Group CAP Documents - "TLS Security," "Certificate Expiry Management," and "OWASP Transport Layer Protection Cheat Sheet" sections.

NEW QUESTION: 15
In the context of NoSQL injection, which of the following is correct?
Statement A: NoSQL databases provide looser consistency restrictions than traditional SQL databases. By requiring fewer relational constraints and consistency checks, NoSQL databases often offer performance and scaling benefits. Yet these databases are still potentially vulnerable to injection attacks, even if they aren't using the traditional SQL syntax.
Statement B: NoSQL database calls are written in the application's programming language, a custom API call, or formatted according to a common convention (such as XML, JSON, LINQ, etc).
A. A is true, and B is false
B. A is false, and B is true
C. Both A and B are false
D. Both A and B are true
Answer: (SHOW ANSWER)

Let's evaluate the two statements about NoSQL injection:
* Statement A: NoSQL databases (e.g., MongoDB, Cassandra) are designed for scalability and flexibility, often sacrificing strict consistency for performance (e.g., eventual consistency in distributed systems). Unlike traditional SQL databases, they do not enforce rigid relational constraints, which simplifies scaling but does not eliminate the risk of injection attacks. Even without SQL syntax, NoSQL databases are vulnerable to injection if user input is not sanitized (e.g., in MongoDB, injecting $where or $ne operators). This statement is true.
* Statement B: NoSQL database queries are typically written in the application's programming language (e.g., JavaScript for MongoDB), using a custom API (e.g., MongoDB's query API), or formatted in standards like JSON, XML, or LINQ. For example, a MongoDB query might look like db.collection.
find({ "key": input }), where input is a JSON-like structure. This statement accurately describes how NoSQL queries are constructed and is true.
* Option A ("A is true, and B is false"): Incorrect, as both statements are true.
* Option B ("A is false, and B is true"): Incorrect, as both statements are true.
* Option C ("Both A and B are false"): Incorrect, as both statements are true.
* Option D ("Both A and B are true"): Correct, as both statements accurately describe NoSQL databases and their vulnerability to injection.
The correct answer is D, aligning with the CAP syllabus under "NoSQL Injection" and "Database Security." References: SecOps Group CAP Documents - "NoSQL Injection Vulnerabilities," "Database Query Security," and "OWASP Top 10 (A03:2021 - Injection)" sections.

NEW QUESTION: 16
Salt is a cryptographically secure random string that is added to a password before it is hashed. In this context, what is the primary objective of salting?
A. To defend against dictionary attacks or attacks against hashed passwords using a rainbow table.
B. To slow down the hash calculation process.
C. To generate a long password hash that is difficult to crack.
D. To add a secret message to the password hash.
Answer: A (LEAVE A REPLY)
Salting is a security technique used in password hashing to enhance protection against specific types of attacks. A salt is a random value added to a password before hashing, ensuring that even if two users have the same password, their hashed outputs will differ. The primary objective of salting is to defend against dictionary attacks and rainbow table attacks. Dictionary attacks involve trying common passwords from a precomputed list, while rainbow table attacks use precomputed tables of hash values to reverse-engineer passwords quickly. By adding a unique salt to each password, the hash becomes unique, rendering precomputed rainbow tables ineffective, as an attacker would need to generate a new table for each salt, which is computationally impractical.

Option B ("To slow down the hash calculation process") is incorrect because while techniques like key stretching (e.g., using PBKDF2 or bcrypt) intentionally slow hashing to counter brute-force attacks, salting itself does not primarily aim to slow the process-it focuses on uniqueness. Option C ("To generate a long password hash that is difficult to crack") is a byproduct of salting but not the primary objective; the length and difficulty come from the hash function and salt combination, not salting alone. Option D ("To add a secret message to the password hash") is incorrect, as a salt is not a secret message but a random value, often stored alongside the hash. This aligns with best practices in authentication security, a key component of the CAP syllabus.
References: SecOps Group CAP Documents - "Secure Coding Practices," "Authentication Security," and
"Cryptographic Techniques" sections.

NEW QUESTION: 17
The DNS entries forwww.ironman.comandwww.hulk.comboth point to the same IP address i.e., 1.3.3.7. How does the web server know which web application is being requested by the end user's browser?
A. The web server inspects the HTTP "Host" header sent by the client.
B. The web server inspects the cookies sent by the client.
C. The web server inspects the client's SSL certificate.
D. The web server uses a reverse DNS lookup of the client's IP address.
Answer: A (LEAVE A REPLY)
When multiple domain names (e.g.,www.ironman.comandwww.hulk.com) resolve to the same IP address (e.
g., 1.3.3.7), a web server hosting multiple applications on that IP must determine which application to serve.
This is achieved using theHTTP "Host" header, which is part of the HTTP/1.1 protocol. The client (browser) includes the requested domain (e.g., Host: www.ironman.com) in the request, allowing the server to route the request to the appropriate virtual host or application configured for that domain. This is a standard practice in virtual hosting.
* Option A ("The web server inspects the HTTP 'Host' header sent by the client"): Correct, as the Host header enables the server to distinguish between applications on the same IP.

* Option B ("The web server inspects the cookies sent by the client"): Incorrect, as cookies are used for session management or personalization, not for identifying the requested application.
* Option C ("The web server inspects the client's SSL certificate"): Incorrect, as SSL certificates are used for encryption and authentication, not for application routing (though they may include the domain name for validation).
* Option D ("The web server uses a reverse DNS lookup of the client's IP address"): Incorrect, as reverse DNS lookup resolves an IP to a domain, which is irrelevant for the server determining the requested application.
The correct answer is A, aligning with the CAP syllabus under "Web Server Configuration" and "HTTP Protocol Security."References: SecOps Group CAP Documents - "HTTP Headers," "Virtual Hosting," and
"OWASP Web Security Testing Guide" sections.

NEW QUESTION: 18
While performing a security audit of a web application, you discovered an exposed docker-compose.yml file.
What is the significance of this file and what data can be found in it?
A. The docker-compose.yml file is a YAML file that contains the application source code.
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. The docker-compose.yml file is a YAML file that is used to define the services, networks, and volumes required for a Docker application. It specifies the configuration and dependencies for all containers in the application, including their network settings and container volumes.
D. The docker-compose.yml file is a YAML file that contains the configuration of load balancers and firewalls.
Answer: (SHOW ANSWER)
A docker-compose.yml file is a YAML-formatted configuration file used with Docker Compose, a tool for defining and running multi-container Docker applications. Its primary significance lies in orchestrating the deployment of Docker containers by specifying services (e.g., web server, database), networks (e.g., internal communication), and volumes (e.g., persistent storage). An exposed docker-compose.yml file poses a security risk because it may reveal sensitive configuration details, such as service names, ports, environment variables (e.g., database credentials), and network settings, which attackers could exploit to target the application.
* Option A ("The docker-compose.yml file is a YAML file that contains the application source code"): Incorrect, as this file defines configuration and orchestration, not source code.
* Option B ("The docker-compose.yml file is a YAML file that contains the server logs and user session information..."): Incorrect, as logs and session data are stored elsewhere (e.g., in container logs or databases), not in docker-compose.yml.
* Option C ("The docker-compose.yml file is a YAML file that is used to define the services, networks, and volumes..."): Correct, as it accurately describes the file's purpose and content, including configuration and dependencies, which are critical for Docker applications.

* Option D ("The docker-compose.yml file is a YAML file that contains the configuration of load balancers and firewalls"): Incorrect, as it focuses only on load balancers and firewalls, which are specific components and not the primary focus of the file.
The correct answer is C, aligning with the CAP syllabus under "Container Security" and "Configuration Management."References: SecOps Group CAP Documents - "Docker Security," "Container Orchestration," and "OWASP Application Security Verification Standard (ASVS)" sections.

NEW QUESTION: 19
The payload {{7*7}} can be used for determining which of the following vulnerabilities?
A. Server Side Template Injection (SSTI)
B. Client-Side Template Injection (CSTI)
C. Both 1 and 2
D. None of the above
Answer: (SHOW ANSWER)
The payload {{7*7}} is a common test string used to detectServer-Side Template Injection (SSTI) vulnerabilities. SSTI occurs when user input is improperly rendered within a server-side template engine (e.g., Jinja2, Freemarker, or Handlebars), allowing the execution of arbitrary template expressions. If the server evaluates {{7*7}} and returns 49 (the result of 7 multiplied by 7), it indicates that the server is processing the input as a template expression, confirming an SSTI vulnerability. This can potentially lead to remote code execution if the template engine supports advanced features.
* Option A ("Server Side Template Injection (SSTI)"): Correct, as {{7*7}} is a standard payload to test for SSTI by checking if the server evaluates the expression.
* Option B ("Client-Side Template Injection (CSTI)"): Incorrect, as CSTI involves client-side rendering (e.g., JavaScript templates like Mustache), and {{7*7}} would not be evaluated on the client unless explicitly designed to do so, which is not implied here.
* Option C ("Both 1 and 2"): Incorrect, as the payload specifically targets server-side processing.
* Option D ("None of the above"): Incorrect, as SSTI is applicable.
The correct answer is A, aligning with the CAP syllabus under "Server-Side Template Injection" and "Input Validation."References: SecOps Group CAP Documents - "SSTI Vulnerabilities," "Template Engine Security," and "OWASP Injection Prevention" sections.

NEW QUESTION: 20
Which of the following is NOT an asymmetric key encryption algorithm?
A. AES
B. RSA
C. Diffie-Hellman
D. DSA
Answer: (SHOW ANSWER)

Asymmetric key encryption (also known as public-key cryptography) uses a pair of keys: a public key for encryption and a private key for decryption (or vice versa for signing). Symmetric key encryption, on the other hand, uses the same key for both encryption and decryption. Let's evaluate the options:
* Option A ("AES"): AES (Advanced Encryption Standard) is a symmetric key encryption algorithm. It uses a single key (e.g., 128, 192, or 256 bits) for both encryption and decryption, making it a symmetric algorithm, not an asymmetric one.
* Option B ("RSA"): RSA (Rivest-Shamir-Adleman) is an asymmetric key encryption algorithm. It uses a public key to encrypt data and a private key to decrypt it, making it a classic example of asymmetric cryptography.
* Option C ("Diffie-Hellman"): Diffie-Hellman is an asymmetric key exchange algorithm. While it is primarily used for key exchange rather than direct encryption, it relies on asymmetric principles (public and private keys) to securely establish a shared secret, so it is considered part of asymmetric cryptography.
* Option D ("DSA"): DSA (Digital Signature Algorithm) is an asymmetric algorithm used for digital signatures. It uses a pair of keys (public and private) for signing and verification, making it an asymmetric algorithm.
The correct answer is A, as AES is the only symmetric algorithm listed, aligning with the CAP syllabus under
"Cryptography Fundamentals" and "Symmetric vs. Asymmetric Encryption."References: SecOps Group CAP Documents - "Cryptographic Algorithms," "Symmetric and Asymmetric Encryption," and "OWASP Cryptographic Storage Cheat Sheet" sections.

NEW QUESTION: 21
In the context of a Dependency Confusion Attack, which of the following files is analyzed for determining potential private packages?
A. package.json
B. requirements.txt
C. Both A and B
D. None of the above
Answer: (SHOW ANSWER)
A Dependency Confusion Attack occurs when an attacker uploads a malicious package with the same name as a private package to a public package repository (e.g., npm, PyPI), causing a package manager to prioritize the malicious public package over the intended private one due to misconfiguration or version precedence. To identify potential private packages for such an attack, attackers analyze dependency configuration files that list the application's dependencies.
* Option A ("package.json"): This file is used by npm (Node.js package manager) to define dependencies for JavaScript projects. It lists package names and versions, making it a prime target for identifying private packages that might be targeted in a dependency confusion attack.

* Option B ("requirements.txt"): This file is used by pip (Python package manager) to define dependencies for Python projects. It similarly lists package names and versions, making it another target for identifying private packages.
* Option C ("Both A and B"): Correct, as both package.json (JavaScript/Node.js) and requirements.txt (Python) are dependency configuration files that can reveal private package names, which an attacker might exploit in a dependency confusion attack.
* Option D ("None of the above"): Incorrect, as both files are relevant.
The correct answer is C, aligning with the CAP syllabus under "Supply Chain Attacks" and "Dependency Management."References: SecOps Group CAP Documents - "Dependency Confusion Attacks," "Software Supply Chain Security," and "OWASP Dependency-Check Guide" sections.

NEW QUESTION: 22
A robots.txt file tells the search engine crawlers about the URLs which the crawler can access on your site.
Which of the following is true about robots.txt?
A. Developers must not list any sensitive files and directories in this file
B. Developers must list all sensitive files and directories in this file to secure them
C. Both A and B
D. None of the above
Answer: (SHOW ANSWER)
The robots.txt file is a text file placed in a website's root directory to communicate with web crawlers (e.g., Googlebot) about which pages or resources should not be accessed or indexed. It uses directives like Disallow to specify restricted areas (e.g., Disallow: /admin/). However, robots.txt is not a security mechanism; it is only a request to crawlers, and malicious bots or users can ignore it.
* Option A ("Developers must not list any sensitive files and directories in this file"): Correct. Listing sensitive files or directories (e.g., Disallow: /secret/) in robots.txt can inadvertently expose their existence to attackers, who can then attempt to access them directly. The best practice is to avoid mentioning sensitive paths and rely on proper access controls (e.g., authentication, authorization) instead.
* Option B ("Developers must list all sensitive files and directories in this file to secure them"): Incorrect. Listing sensitive paths in robots.txt does not secure them; it only informs crawlers to avoid them, and it can serve as a roadmap for attackers.
* Option C ("Both A and B"): Incorrect, as A and B are contradictory; B is false.
* Option D ("None of the above"): Incorrect, as A is true.
The correct answer is A, aligning with the CAP syllabus under "Web Crawler Security" and "Information Disclosure Prevention."References: SecOps Group CAP Documents - "robots.txt Usage," "Information Exposure," and "OWASP Web Security Testing Guide" sections.

NEW QUESTION: 23

Which of the following attributes is NOT used to secure the cookie?

A. HttpOnly

B. Secure

C. Restrict

D. Same-Site

Answer: (SHOW ANSWER)

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."References: SecOps Group CAP Documents - "Cookie Security Attributes," "Session Security," and "OWASP Session Management Cheat Sheet" sections.

NEW QUESTION: 24

Which of the following is correct?

A. The browser contains the private key of all known Certifying Authorities (CA) and based on that, it differentiates between a valid and an invalid TLS Certificate

B. The browser contains the public key of all known Certifying Authorities (CA) and based on that it is able to differentiate between a valid and an invalid TLS Certificate

C. 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

D. The browser does not have any mechanism to validate the TLS Certificate

Answer: (SHOW ANSWER)

TLS (Transport Layer Security) certificates are validated by browsers to ensure secure communication.

Browsers maintain a trusted store ofpublic keysfrom known Certifying Authorities (CAs), which are used to verify the digital signature of a TLS certificate presented by a server. This process

involves checking the certificate's signature against the CA's public key to confirm its authenticity and validity. If the signature matches and other criteria (e.g., expiration, revocation) are met, the certificate is deemed valid.
* Option A ("The browser contains the private key..."): Incorrect, as browsers do not contain private keys of CAs; private keys are kept secret by the CAs themselves.
* Option B ("The browser contains the public key..."): Correct, as browsers use CA publickeys to validate certificates, enabling differentiation between valid and invalid TLS certificates.
* Option C ("The browser contains both the public and private key..."): Incorrect, as browsers only store public keys, not private keys, for security reasons.
* Option D ("The browser does not have any mechanism..."): Incorrect, as browsers have robust mechanisms (via CA public keys) to validate TLS certificates.
The correct answer is B, aligning with the CAP syllabus under "Secure Communication" and "TLS Configuration."References: SecOps Group CAP Documents - "TLS/SSL Security," "Certificate Validation," and "OWASP Cryptographic Practices" sections.

NEW QUESTION: 25
A website administrator forgot to renew the TLS certificate on time and as a result, the application is now displaying a TLS error message. However, on closer inspection, it appears that the error is due to the TLS certificate expiry.
Which of the following is correct?
A. There is no urgency to renew the certificate as the communication is still over TLS
B. There is an urgency to renew the certificate as the users of the website may get conditioned to ignore TLS warnings and therefore ignore a legitimate warning which could be a real Man-in-the-Middle attack
Answer: (SHOW ANSWER)
A TLS certificate expiry means the certificate used to secure the HTTPS connection is no longer valid, typically due to its expiration date being passed. This triggers a TLS error message in the browser (e.g., "Your connection is not private"). Let's evaluate the options:
* Option A ("There is no urgency to renew the certificate as the communication is still over TLS"): Incorrect. While the communication may technically still occur over TLS (depending on browser and server behavior), an expired certificate breaks the trust model. Browsers will warn users, and some may block access entirely. The communication is not secure in the sense that the certificate's validity cannot be verified, potentially exposing users to risks if they bypass warnings. This is not a valid justification for delaying renewal.
* Option B ("There is an urgency to renew the certificate as the users of the website may get conditioned to ignore TLS warnings and therefore ignore a legitimate warning which could be a real Man-in-the-Middle attack"): Correct. An expired TLS certificate causes repeated warnings, which may desensitize users to ignore them. If a real Man-in-the-Middle (MitM) attack occurs (e.g., an attacker presents a fake certificate), users accustomed to bypassing warnings might not notice, increasing the risk of data interception. Renewing the certificate is urgent to restore trust and prevent this conditioning effect, aligning with security best practices.

The correct answer is B, aligning with the CAP syllabus under "TLS Configuration" and "Certificate Management."References: SecOps Group CAP Documents - "TLS Security," "Certificate Expiry Management," and "OWASP Transport Layer Protection Cheat Sheet" sections.

NEW QUESTION: 26

In the screenshot below, an attacker is attempting to exploit which vulnerability?

POST /upload.php HTTP/1.1
Host: example.com
Cookie: session=xyz123;JSESSIONID=abc123
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) rv:107.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW Content-Length: 12345
Connection: keep-alive Content-Disposition: form-data; name="avatar"; filename="malicious.php"
Content-Type: image/jpeg
<?php
phpinfo();
?>

A. HTTP Desync Attack
B. File Path Traversal Attack
C. File Upload Vulnerability
D. Server-Side Request Forgery

Answer: (SHOW ANSWER)

The screenshot shows an HTTP POST request to /upload.php with a multipart/form-data payload, where the attacker uploads a file named malicious.php disguised as an image/jpeg but containing PHP code (<?php phpinfo(); ?>). This indicates an attempt to exploit aFile Upload Vulnerability. Such vulnerabilities occur when an application allows users to upload files without proper validation or sanitization, enabling attackers to upload malicious scripts (e.g., PHP) that can be executed on the server. In this case, if the server executes the uploaded malicious.php, it could expose server information via phpinfo() or perform other malicious actions.
Option A ("HTTP Desync Attack") involves manipulating HTTP request pipelines, which is not relevant here as the request appears standard. Option B ("File Path Traversal Attack") involves accessing unauthorized files using ../, which is not evident in this request. Option D ("Server-Side Request Forgery") involves tricking the server into making unintended requests, which does not apply to file uploads. Thus, C is the correct answer, aligning with the CAP syllabus under "File Handling Security" and "OWASP Top 10 (A05:2021 - Security Misconfiguration)."References: SecOps Group CAP Documents - "File Upload Vulnerabilities," "Input Validation," and "OWASP Top 10" sections.

NEW QUESTION: 27

Based on the screenshot above, which of the following is the most true?
Screenshot
![Login Form]
coder@viewer
User does not exist
[Password field]
Forget password?
[Login button]
Not yet member? Sign now
A. The application is vulnerable to username enumeration
B. The application is vulnerable to brute-force attacks
C. The application does not enforce a strong password policy
D. None of the above
Answer: (SHOW ANSWER)
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."References: SecOps Group CAP Documents - "Authentication Best Practices," "Enumeration Attacks," and "OWASP Authentication Cheat Sheet" sections.