ITCERTKING

The latest IT certification exam materials

Exam : **CAP-JPN**

Title : CAP - Certified Authorization Professional (CAP日本語版)

Vendor : ISC

Version : DEMO

QUESTION NO: 1
以下のスクリーンショットでは、攻撃者はどの脆弱性を悪用しようとしていますか?
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 非同期攻撃
B. ファイルパストラバーサル攻撃
C. ファイルアップロードの脆弱性
D. サーバー側リクエストフォージェリ
Answer: C
Explanation:
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.

QUESTION NO: 2
MySQL データベースの SQL
インジェクション脆弱性を手動で悪用する際に、ファイルの内容を読み取るために使用できる SQL 関数はどれですか?
A. READ_FILE()
B. LOAD_FILE()
C. FETCH_FILE()

D. GET_FILE()
Answer: B
Explanation:
SQL injection vulnerabilities allow attackers to manipulate database queries, potentially accessing unauthorized data, including file contents, if the database supports such operations. In MySQL, the LOAD_FILE()function is specifically designed to read the contents of a file on the server where the database is hosted, provided the file exists, the database user has appropriate privileges (e.g., FILE privilege), and the file is readable. For example, SELECT LOAD_FILE('/etc/passwd') could extract the contents of the /etc
/passwd file if exploitable.
* Option A ("READ_FILE()"): This is not a valid MySQL function.
* Option B ("LOAD_FILE()"): This is the correct function for reading file contents in MySQL, making it the right choice for exploitation.
* Option C ("FETCH_FILE()"): This is not a recognized MySQL function.
* Option D ("GET_FILE()"): This is also not a valid MySQL function.
The correct answer is B, aligning with the CAP syllabus under "SQL Injection" and "Database Security." References: SecOps Group CAP Documents - "Injection Vulnerabilities," "MySQL Security Features," and
"OWASP Top 10 (A03:2021 - Injection)" sections.

QUESTION NO: 3
以下のリクエスト/レスポンスに基づいて、次の記述のうち正しいものはどれですか?
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. アプリケーションはオープンリダイレクトの脆弱性の影響を受ける可能性があります
B. アプリケーションはクロスサイトリクエストフォージェリの脆弱性に対して脆弱です
C. アプリケーションは安全でないプロトコルを使用しています
D. 上記のすべて
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."References: SecOps Group CAP Documents - "Open Redirection," "Input Validation for Redirects," and "OWASP Top 10 (A10:2021 - Server-Side Request Forgery)" sections.

QUESTION NO: 4
GraphQL は、API 用のオープンソースのデータ クエリおよび操作言語であり、クエリ ランタイム エンジンです。この文脈では、GraphQL イントロスペクションとは何でしょうか?
A. GraphQL APIと他のシステムとの互換性をテストする手法
B. GraphQL APIのパフォーマンスをテストする手法
C. GraphQL APIの構造を発見する手法
D. GraphQL APIのセキュリティをテストする手法
Answer: C
Explanation:
GraphQL Introspection is a built-in feature of GraphQL that allows clients to query the schema of a GraphQL API at runtime. This process involves sending introspection queries (e.g., __schema or __type) to retrieve information about the API's structure, including available types, fields, queries, mutations, and their relationships. This capability is powerful for developers to explore and document APIs but poses a security risk if left enabled in production, as attackers can use it to map out the entire API structure and identify potential attack vectors.
* Option A ("A technique for testing the compatibility of the GraphQL API with other systems"):

Incorrect, as introspection is about schema discovery, not compatibility testing.
* Option B ("A technique for testing the performance of the GraphQL API"): Incorrect, as performance testing involves load or stress testing, not schema exploration.
* Option C ("A technique for discovering the structure of the GraphQL API"): Correct, as introspection is specifically designed to expose the API's schema and structure.
* Option D ("A technique for testing the security of the GraphQL API"): Incorrect, as security testing is a separate process; introspection itself is a feature, not a security test.
The correct answer is C, aligning with the CAP syllabus under "GraphQL Security" and "API Introspection." References: SecOps Group CAP Documents - "GraphQL Fundamentals," "Introspection Risks," and
"OWASP API Security Top 10" sections.

QUESTION NO: 5
CORS (クロスオリジン リソース共有)
の誤った構成に関して、次の記述のうち正しいものはどれですか。
A. HTTP ヘッダーの値が Access-Control-Allow-Origin: * かつ Access-Control-Allow-Credentials: true の場合、CORS が悪用される可能性があります。
B. HTTP ヘッダーの値が Access-Control-Allow-Origin: * かつ Access-Control-Allow-Credentials: false の場合、CORS は悪用される可能性があります。
C. HTTP ヘッダーの値が Access-Control-Allow-Origin: * であり、Access-Control-Allow-Credentials ヘッダーの値が無関係である場合、CORS は悪用される可能性があります。
D. 上記のすべて
Answer: A
Explanation:
CORS (Cross-Origin Resource Sharing) is a mechanism that allows servers to specify which origins can access their resources, enhancing security for cross-origin requests. A common misconfiguration occurs with theAccess-Control-Allow-OriginandAccess-Control-Allow-Credentialsheaders. WhenAccess-Control- Allow-Originis set to * (wildcard, allowing all origins), it permits any domain to make requests. However, if Access-Control-Allow-Credentialsis set to true (allowing credentials like cookies or HTTP authentication), this creates a security risk. Browsers will block such requests because sending credentials with a wildcard origin violates CORS security policies, but an attacker could exploit this misconfiguration to trick a victim's browser into making unauthorized requests if other controls are absent.
Option A is correct because the combination of Access-Control-Allow-Origin: * and Access-Control-Allow- Credentials: true is exploitable, as it enables potential credential leakage or unauthorized access. Option B is incorrect because Access-Control-Allow-Credentials: false disables credential sending, reducing exploitability. Option C is incorrect because the value of Access-Control-Allow-Credentials is not irrelevant; it must be false with a wildcard origin to comply with security standards. Option D ("All of the above") is incorrect as only A holds true. This is a key topic in the CAP syllabus under "CORS Misconfiguration" and
"Client-Side Security."References: SecOps Group CAP Documents - "CORS Configuration," "Security Misconfigurations," and "OWASP Secure Headers" sections.

QUESTION NO: 6

次のどれが非対称鍵暗号化アルゴリズムではありませんか？
A. AES
B. RSA
C. ディフィー・ヘルマン
D. DSA
Answer: A
Explanation:
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.