

---

**Exam : ISA-CAP**

**Title : ISA Certified Automation  
Professional (CAP)**

**Version : DEMO**

---

1. The DNS entries for `www.ironman.com` and `www.hulk.com` both 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**

**Explanation:**

When multiple domain names (e.g., `www.ironman.com` and `www.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 the HTTP "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.

2. 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`.

There are two fixes:

(Bob's Fix): In order to fix this vulnerability, a developer called Bob devised a fix so that the URL does not disclose the numeric value of the `order_id` but uses a SHA1 hash of the `order_id` in the URL, such as:

`https://example.com/?order_id=1ff0fe6f1599536d1326418124a261bc98b8ea1`

Note: that the SHA1 value of 53870 is 1ff0fe6f1599536d1326418124a261bc98b8ea1

(John's Fix): Another developer called John devised a different fix so that the URL does not disclose the numeric value of the `order_id` and uses a Base64 encoded value of the `order_id` in the URL, such as:

`https://example.com/?order_id=NTM4NzA=`

Note: that the Base64 encoded value of 53870 is NTM4NzA=

Which of the following is correct?

- 
- A. Both solutions are adequate to fix the problem
  - B. Both solutions are inadequate and the vulnerability is still not fixed
  - C. Only John's solution fixes the problem
  - D. Only Bob's solution fixes the problem

**Answer: B**

**Explanation:**

The vulnerability described is an Insecure Direct Object Reference (IDOR), where manipulating the `order_id` (e.g., 53870) allows unauthorized access to other users' orders.

The fixes proposed by Bob and John aim to obscure the numeric value of `order_id` to prevent easy guessing or manipulation:

**Bob's Fix (SHA1 Hash):** Replaces `order_id=53870` with `order_id=1ff0fe6f1599536d1326418124a261bc98b8ea1` (SHA1 hash of 53870). While this obscures the original value, an attacker can still attempt to hash potential order IDs (e.g., 53871, 53872) and test them in the URL. If the application directly uses the hash to look up the order without validating the user's authorization, the vulnerability persists. SHA1 is a one-way hash, but it does not inherently enforce access control.

**John's Fix (Base64 Encoding):** Replaces `order_id=53870` with `order_id=NTM4NzA=` (Base64 encoding of 53870). Base64 is a reversible encoding, and an attacker can easily decode `NTM4NzA=` back to 53870 using standard tools. If the application decodes it and uses the original value to fetch orders without authorization checks, the IDOR vulnerability remains.

**Evaluation:** Both fixes address the symptom (disclosing the numeric value) but fail to address the root cause: lack of authorization validation. The application must ensure that only the authenticated user can access their own orders, regardless of the `order_id` format (numeric, hashed, or encoded). Neither fix includes such a check, so the vulnerability persists.

**Option A ("Both solutions are adequate to fix the problem"):** Incorrect, as neither solution enforces authorization.

**Option B ("Both solutions are inadequate and the vulnerability is still not fixed"):** Correct, as both SHA1 hashing and Base64 encoding are superficial changes that do not prevent unauthorized access.

**Option C ("Only John's solution fixes the problem"):** Incorrect, as John's Base64 encoding is reversible and does not fix the IDOR issue.

**Option D ("Only Bob's solution fixes the problem"):** Incorrect, as Bob's SHA1 hashing also does not address the authorization flaw.

The correct answer is B, aligning with the CAP syllabus under "Insecure Direct Object References (IDOR)" and "Access Control Best Practices." References: SecOps Group CAP Documents - "IDOR Mitigation," "Cryptographic Hashing," and "OWASP Access Control Testing Guide" sections.

3. 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: D**

**Explanation:**

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.

4. Under the same-origin policy (also SOP), a web browser permits scripts contained in a web page to access data in another web page, but only if both web pages have the same origin.

Which of the following pages are in the same origin as that of the below URL?

<http://www.example.com/dir/page2.html>

<http://www.example.com/dir/other.html>

<http://www.example.com:81/dir/other.html>

<http://www.example.com/dir/other.html>

<http://en.example.com/dir/other.html>

A. 1 Only

B. 1 and 2

C. 1, 3 and 4

D. None of the above

**Answer: A**

**Explanation:**

The Same-Origin Policy (SOP) is a fundamental security mechanism in web browsers that restricts how a document or script loaded from one origin can interact with resources from another origin. An origin is defined by the combination of protocol, host, and port.

Two pages have the same origin if their protocol (e.g., http://), host (e.g., www.example.com), and port (default is 80 for HTTP) match exactly. The given URL is

<http://www.example.com/dir/page2.html>, which uses HTTP, host www.example.com, and the default port 80.

Option 1 ("<http://www.example.com/dir/other.html>"): Matches the protocol (http), host (www.example.com), and port (default 80), so it is in the same origin.

Option 2 ("<http://www.example.com:81/dir/other.html>"): Differs in port (81 vs. default 80), so it is a

---

different origin.

Option 3 ("http://www.example.com/dir/other.html"): Identical to Option 1, matching protocol, host, and port, so it is in the same origin. However, since it's a duplicate of Option 1 in the list, it doesn't add a new page.

Option 4 ("http://en.example.com/dir/other.html"): Differs in host (en.example.com vs. www.example.com), so it is a different origin.

Since the question asks which pages are in the same origin, and only Option 1 (and its duplicate Option 3) matches, the correct answer is A ("1 Only"), considering unique pages. The CAP syllabus covers SOP under "Client-Side Security" and "Cross-Origin Resource Sharing (CORS)." References: SecOps Group CAP Documents - "Same-Origin Policy," "Browser Security Models," and "OWASP Client-Side Security" sections.

5. An application's forgot password functionality is described below:

The user enters their email address and receives a message on the web page:

"If the email exists, we will email you a link to reset the password"

The user also receives an email saying:

"Please use the link below to create a new password:"

(Note that the developer has included a one-time random token with the 'userId' parameter in the link).

So, the link seems like:

[https://example.com/reset\\_password?userId=5298&token=70e7803e-bf53-45e1-8a3f-fb15da7de3a0](https://example.com/reset_password?userId=5298&token=70e7803e-bf53-45e1-8a3f-fb15da7de3a0)

Will this mechanism prevent an attacker from resetting arbitrary users' passwords?

A. True

B. False

**Answer: A**

**Explanation:**

The password reset mechanism now includes a one-time random token alongside the userId in the reset link: [https://example.com/reset\\_password?userId=5298&token=70e7803e-bf53-45e1-8a3f-fb15da7de3a0](https://example.com/reset_password?userId=5298&token=70e7803e-bf53-45e1-8a3f-fb15da7de3a0).

Let's evaluate the effectiveness of this mechanism:

Analysis: The inclusion of a one-time random token (e.g., 70e7803e-bf53-45e1-8a3f-fb15da7de3a0) is a best practice for secure password reset mechanisms. This token should be unique, unpredictable, and tied to the specific user's reset request (e.g., stored in the database with an expiration time). When the user clicks the link, the application should verify that the token matches the one associated with the userId and that it hasn't been used or expired.

Preventing Arbitrary Resets: Without the token, an attacker could manipulate the userId (e.g., to 5299) to reset another user's password (as seen in Question 45). However, with the token, the attacker would need to guess the correct token for the targeted userId, which is computationally infeasible if the token is sufficiently random (e.g., a UUID or cryptographically secure random string) and properly validated. This prevents unauthorized password resets.

Additional Considerations: The link uses https://, ensuring a secure channel (unlike Question 45). The message "If the email exists..." prevents username enumeration, as discussed previously.

Option A ("True"): Correct, as the one-time random token, if implemented correctly (e.g., validated server-side, single-use, time-limited), prevents an attacker from resetting arbitrary users' passwords.

Option B ("False"): Incorrect, as the mechanism is secure with the token.

---

The correct answer is A, aligning with the CAP syllabus under "Secure Password Reset" and "Token-Based Authentication." References: SecOps Group CAP Documents - "Password Reset Security," "Token Generation Best Practices," and "OWASP Forgot Password Cheat Sheet" sections.