CERTLABZ CERTIFICATION BOARD

Certified Secure Development Specialist (CSDS)

Ten Sample Questions

These ten questions are written at the difficulty of the live exam. Six carry a figure, a data table, a configuration or a code listing; four are plain scenario questions, as on the live paper. Every one is a scenario a working secure development specialist would meet, and none of them appears in the live question bank.

The live exam is multistage adaptive: how you score on one stage selects the difficulty of the next, so no two candidates sit the same paper. It is a performance-based exam: each form can carry up to 5 to 12 performance-based questions alongside the multiple-choice questions, which means you build, sequence or complete something rather than only choosing from a list. Answers and full explanations follow each question here; the live exam does not disclose item-level answers.

Question 1 — Domain 1: Secure Development Foundations — DATA

Sarah Miller reviews how much access each service account holds.

Account	Access
Report reader	Full admin
Backup job	Full admin
Web app	Full admin
Health check	Full admin

Table 1.1 Access granted today.

Which principle should guide the fix?

A. Open by default
B. Least privilege
C. Trust all clients
D. Security by obscurity

Correct answer: B
Every account holds full admin rights it does not need, which widens the damage from any single compromise. Least privilege grants each account only the access its job requires, so a breached report reader cannot touch backups or production. The other choices increase exposure rather than reduce it.

Question 2 Domain 2: Common Vulnerabilities CONFIG

James Carter reads a query that a request builds from user input.

```
query = "SELECT * FROM users " +
        "WHERE name = " + input;
db.run(query);  // input unchecked
```

Listing 2.1 A query built by string joins.

Which flaw must be fixed here?

A Clickjacking
B DNS caching
C SQL injection
D Slow log rotation

Correct answer: C
Joining raw user input into a query lets an attacker change the query and read or alter data, which is SQL injection. The fix is parameterised queries or prepared statements that keep data separate from code. The other options describe unrelated concerns that this snippet does not raise.

Question 3 Domain 3: Secure Design and Coding CONFIG

Olivia Davis reviews code that writes user text straight into a page.

```
page = "<div>" + userText + "</div>";
response.send(page);  // not escaped
```

Listing 3.1 User text placed into HTML.

What best prevents the flaw?

A Output encoding
B Bigger server
C More comments
D Longer variable names

Correct answer: A
Placing unescaped user text into HTML lets a script in that text run in the browser, which is cross-site scripting. Encoding the output for its context neutralises the markup so it renders as plain text. Server size, comments and variable names have no effect on this flaw.

Question 4 — Domain 4: Authentication, Authorization and Secrets — DATA

David Brown records where each secret lives in the current build.

Secret	Location
DB password	In source code
API key	Committed to Git
Token	In server logs
Cert key	In source code

Table 4.1 Where secrets live now.

What is the soundest fix?

A. Print to server logs
B. Hardcode in source
C. Commit to Git repo
D. Use a secrets vault

Correct answer: D
Secrets in source code, version control or logs are exposed to anyone with read access and are hard to rotate. A managed secrets vault stores them outside the code, controls who can read them and supports rotation. The other options repeat the very practices that leaked the secrets.

Question 5 — Domain 5: Testing and Verification — DATA

Liam Anderson matches each test type to what it catches.

Test	What it catches
SAST	Code flaws
DAST	Runtime flaws
SCA	(to choose)
Unit test	Logic errors

Table 5.1 Test types and targets.

Which activity fills the SCA row?

A. Manual code reformatting
B. Dependency scanning

C Uptime monitor

D Log color theme

Correct answer: B
Software composition analysis inspects third-party libraries and flags known vulnerable versions, so dependency scanning is its job. This catches risk that code-only tools miss because the flaw lives in imported packages. Reformatting, uptime and log styling do nothing to surface vulnerable dependencies.

Question 6

Domain 6: DevSecOps and Operations

CONFIG

Ava Thompson reviews a deploy pipeline that leaks a key and skips checks.

```
deploy:
  echo $API_KEY  # printed to log
  skip: security scan
  push: production
```

Listing 6.1 The current pipeline.

What should change first?

A Stop logging secrets

B Log more secrets

C Remove all tests

D Deploy without review

Correct answer: A
Echoing the key writes a live secret into build logs that many people can read, which is the most immediate exposure. The first change is to stop printing secrets and inject them from a protected store, then restore the security scan. The other options make the pipeline less safe, not more.

Question 7

Domain 1: Secure Development Foundations

A team wants to find design flaws before any code is written. Which practice fits best?

A Ship then patch

B Skip design docs

C Threat modeling

D Guess at the risks

Correct answer: C
Threat modeling examines the design to find where an attacker could act and what could go wrong, before code exists to change. Doing it early is far cheaper than patching a shipped flaw. Skipping design work or guessing leaves the same weaknesses in place.

Question 8 Domain 2: Common Vulnerabilities

A download endpoint reads whatever file path the user sends. What prevents abuse?

A. Trust the user input
B. Read any path given
C. Return every system file
D. Use a path allowlist

Correct answer: D
Reading an unchecked path lets a user request files outside the intended folder, which is path traversal. Confining requests to an allowlist of permitted files or a fixed safe directory blocks access to everything else. Trusting the input or reading any path is exactly the weakness being exploited.

Question 9 Domain 4: Authentication, Authorization and Secrets

A service needs to store user passwords. How should they be kept?

A. As reversible base64
B. Salted and hashed
C. As plain text
D. In a shared config

Correct answer: B
Passwords should be stored using a strong, salted one-way hash so the raw value is never recoverable from the store. A per-user salt stops precomputed lookups from matching many accounts at once. Plain text, base64 and shared config all leave the real passwords readable to anyone with access.

Question 10 Domain 5: Testing and Verification

How should security tests run so flaws are caught as early as possible?

A. Automatically in CI
B. Manually once a year
C. Only after a breach
D. Never once it ships

Correct answer: A
Running security tests automatically in the pipeline checks every change as it lands, so flaws are caught before release. Annual manual passes or post-breach reviews find problems long after they reach production. Not testing after launch lets regressions slip through unnoticed.

Published by the CertLabz Certification Board, a division of CertLabz LLC. Companion to objectives document CSDS-001. https://certlabz.com/certifications/certified-secure-development-specialist.html