GitHub Advanced Security Certification Pracice Test (Sample)

Study Guide

Everything you need from our exam experts!

Table of Contents

Introduction

Preparing for a certification exam can feel overwhelming, but with the right tools, it becomes an opportunity to build confidence, sharpen your skills, and move one step closer to your goals. At Examzify, we believe that effective exam preparation isn't just about memorization, it's about understanding the material, identifying knowledge gaps, and building the test-taking strategies that lead to success.

This guide was designed to help you do exactly that.

Whether you're preparing for a licensing exam, professional certification, or entry-level qualification, this book offers structured practice to reinforce key concepts. You'll find a wide range of multiple-choice questions, each followed by clear explanations to help you understand not just the right answer, but why it's correct.

The content in this guide is based on real-world exam objectives and aligned with the types of questions and topics commonly found on official tests. It's ideal for learners who want to:

- **Practice answering questions under realistic conditions,**
- **Improve accuracy and speed,**
- **Review explanations to strengthen weak areas, and**
- **Approach the exam with greater confidence.**

We recommend using this book not as a stand-alone study tool, but alongside other resources like flashcards, textbooks, or hands-on training. For best results, we recommend working through each question, reflecting on the explanation provided, and revisiting the topics that challenge you most.

Remember: successful test preparation isn't about getting every question right the first time, it's about learning from your mistakes and improving over time. Stay focused, trust the process, and know that every page you turn brings you closer to success.

Let's begin.

How to Use This Guide

This guide is designed to help you study more effectively and approach your exam with confidence. Whether you're reviewing for the first time or doing a final refresh, here’s how to get the most out of your Examzify study guide:

1. Start with a Diagnostic Review

Skim through the questions to get a sense of what you know and what you need to focus on. Your goal is to identify knowledge gaps early.

2. Study in Short, Focused Sessions

Break your study time into manageable blocks (e.g. 30 – 45 minutes). Review a handful of questions, reflect on the explanations.

3. Learn from the Explanations

After answering a question, always read the explanation, even if you got it right. It reinforces key points, corrects misunderstandings, and teaches subtle distinctions between similar answers.

4. Track Your Progress

Use bookmarks or notes (if reading digitally) to mark difficult questions. Revisit these regularly and track improvements over time.

5. Simulate the Real Exam

Once you're comfortable, try taking a full set of questions without pausing. Set a timer and simulate test-day conditions to build confidence and time management skills.

6. Repeat and Review

Don’t just study once, repetition builds retention. Re-attempt questions after a few days and revisit explanations to reinforce learning. Pair this guide with other Examzify tools like flashcards, and digital practice tests to strengthen your preparation across formats.

There’s no single right way to study, but consistent, thoughtful effort always wins. Use this guide flexibly, adapt the tips above to fit your pace and learning style. You've got this!

Questions

1. What is Dependency Graph, and why is it required for Dependency Review and Dependabot?

A. The Dependency Graph maps a repository's dependencies; it is required to identify vulnerable components and suggest updates.

B. It tracks issues by label

C. It lists all contributors

D. It is optional and only used for license scanning

2. In GitHub Advanced Security, what is an advisory and how does Dependabot use advisories?

A. It records a vulnerability; Dependabot uses advisories to determine fixes and create updates.

B. It defines user roles for security responders within the repo.

C. It stores policy guidelines for incident response.

D. It tracks performance metrics of security scans.

3. How can you export Code Scanning results for integration with external tools?

A. Export or download SARIF results from the Security tab or workflow artifacts to feed into SIEMs or scanners

B. Copy results by hand from the UI

C. Email the results to the security team

D. Re-run Code Scanning in a separate tool

4. Explain the difference between a Code Scanning alert and a Dependabot alert.

A. Code Scanning alerts indicate missing documentation.

B. Code Scanning alerts flag static analysis findings in code; Dependabot alerts inform about vulnerable dependencies in the project.

C. Code Scanning alerts point to vulnerability in dependencies; Dependabot alerts point to code vulnerabilities.

D. Both alerts are the same concept.

5. Which of the following statements about SBOM is most accurate?

A. SBOM encrypts dependencies to prevent tampering.

B. SBOM tracks license terms for each dependency.

C. SBOM provides visibility into every component and dependency to assess risk and compliance.

D. SBOM manages user access to dependencies.

6. What is the difference between Direct dependencies and Transitive dependencies in the context of Dependency Review?

A. Direct dependencies are explicitly listed in your manifest; transitive dependencies are pulled in by those direct dependencies.

B. Direct dependencies are pulled in by transitive.

C. Transitive dependencies are explicitly listed.

D. Direct dependencies are not in manifest.

7. After Phase 1, which phase follows?

A. Phase 2

B. Phase 0

C. Phase 3

D. Phase 1

8. What does git filter-repo do?

A. It converts repository formats.

B. It creates backups automatically.

C. It removes passwords or renaming of specific files or file sets.

D. It integrates with issue trackers.

9. How many languages does CodeQL support in GAS and how do you enable them?

A. Multiple languages (e.g., Java, JavaScript/TypeScript, Python, C/C++, Go); configure the codeql-analysis workflow to enable relevant languages

B. Only Java and JavaScript

C. It supports only one language per workflow

D. It cannot be configured; it auto-detects languages

10. What is the process for triaging a Code Scanning alert?

A. Open the alert, review location and rule, determine severity, reproduce if needed, assign to a developer, and decide on remediation or dismissal.

B. Wait for automatic escalation by system.

C. Delete the alert and ignore.

D. Submit a pull request to remove the rule.

Answers

1. A
2. A
3. A
4. B
5. C
6. A
7. A
8. C
9. A
10. A

Explanations

1. What is Dependency Graph, and why is it required for Dependency Review and Dependabot?

A. The Dependency Graph maps a repository's dependencies; it is required to identify vulnerable components and suggest updates.

B. It tracks issues by label

C. It lists all contributors

D. It is optional and only used for license scanning

Dependency Graph maps a repository's dependencies, showing direct and transitive relationships and the versions involved. This is what enables Dependency Review to see exactly what would be added or changed in a PR and to surface any security or licensing implications tied to those dependencies. It also powers Dependabot by providing a complete dependency tree that can be checked against vulnerability databases and used to propose compatible, safe updates. Without this graph, the service wouldn't know which components exist, how they connect, or how updates propagate through the tree, so vulnerability checks and update suggestions wouldn't be reliable.

2. In GitHub Advanced Security, what is an advisory and how does Dependabot use advisories?

A. It records a vulnerability; Dependabot uses advisories to determine fixes and create updates.

B. It defines user roles for security responders within the repo.

C. It stores policy guidelines for incident response.

D. It tracks performance metrics of security scans.

Advisories are formal vulnerability records published in GitHub's Security Advisory Database. They describe a flaw in a package, which versions are affected, which versions fix it, and often include references and severity information. Dependabot uses these advisories as the source of truth for vulnerabilities in your dependencies. When a dependency in your project matches an advisory's vulnerable range, Dependabot uses the advisory to determine the appropriate fixed version and then creates automated updates (pull requests) to upgrade to a non-vulnerable version. This is why advisories exist: they provide the vulnerability details and the remediation path that tools like Dependabot can act on.

3. How can you export Code Scanning results for integration with external tools?

A. Export or download SARIF results from the Security tab or workflow artifacts to feed into SIEMs or scanners

B. Copy results by hand from the UI

C. Email the results to the security team

D. Re-run Code Scanning in a separate tool

To integrate Code Scanning results with external tools, you want a portable, machine-readable output that external systems can ingest automatically. The recommended path is to export or download SARIF results from the Security tab or as a workflow artifact. SARIF, the Static Analysis Results Interoperability Format, is a standardized representation of findings that many SIEMs, scanners, and security platforms can parse. By obtaining a SARIF file, you enable automated ingestion, consistent formatting of issue details (locations, severities, rules), and smooth integration into your broader security workflows. You can grab the SARIF file directly from the UI for a given run, or configure your CI/CD workflow to produce and publish the SARIF artifact as part of Code Scanning, ensuring reproducibility and easy automation. Copying results by hand isn't scalable and prone to human error, emailing results isn't machine-readable or automatable, and re-running Code Scanning in a separate tool doesn't produce a portable export from GitHub to feed into external systems.

4. Explain the difference between a Code Scanning alert and a Dependabot alert.

A. Code Scanning alerts indicate missing documentation.

B. Code Scanning alerts flag static analysis findings in code; Dependabot alerts inform about vulnerable dependencies in the project.

C. Code Scanning alerts point to vulnerability in dependencies; Dependabot alerts point to code vulnerabilities.

D. Both alerts are the same concept.

The difference being tested is what each alert monitors and reports on. Code Scanning alerts come from static analysis of your source code, looking for insecure coding patterns, potential vulnerabilities, or risky configurations inside the codebase itself. They analyze the code you write (and sometimes its configuration) to identify issues such as unsafe patterns or logic flaws that could lead to security problems. Dependabot alerts, on the other hand, focus on your project's dependencies. They track known vulnerabilities in libraries and packages your project uses, including transitive dependencies, and notify you when a dependency has a published security advisory so you can update to a safe version. So the best answer reflects that distinction: Code Scanning flags findings in the code through static analysis, while Dependabot flags vulnerabilities in the dependencies.

5. Which of the following statements about SBOM is most accurate?

A. SBOM encrypts dependencies to prevent tampering.

B. SBOM tracks license terms for each dependency.

C. SBOM provides visibility into every component and dependency to assess risk and compliance.

D. SBOM manages user access to dependencies.

SBOM centers on inventorying all components and dependencies in a piece of software so you know exactly what's inside and where it came from. That visibility is what enables you to assess risk and compliance across the software supply chain. You can identify which open-source libraries and third-party components are included, check for known vulnerabilities, track licensing obligations, and verify provenance. Encrypting dependencies would hide information and defeat the purpose of visibility. While an SBOM may carry license information as part of its data, its main value is not simply tracking licenses but giving a complete view of components for risk and compliance. It also doesn't handle who can access dependencies—that's a separate access-control concern.

6. What is the difference between Direct dependencies and Transitive dependencies in the context of Dependency Review?

A. Direct dependencies are explicitly listed in your manifest; transitive dependencies are pulled in by those direct dependencies.

B. Direct dependencies are pulled in by transitive.

C. Transitive dependencies are explicitly listed.

D. Direct dependencies are not in manifest.

Direct dependencies are the ones you explicitly declare in your project's manifest, such as in package.json or pom.xml. Transitive dependencies are the dependencies brought in indirectly through those direct dependencies—the dependencies of your dependencies. In Dependency Review, this distinction matters because you control and review what you directly include, while the transitive ones are pulled in by those direct choices and may carry their own security or licensing implications. For example, if you declare a library A, and A depends on B and C, then B and C are transitive dependencies. The idea that direct dependencies are pulled in by transitive is the reverse, and the notion that transitive dependencies are explicitly listed is generally incorrect since they're not usually declared by you but provided by the direct dependencies. Direct dependencies are indeed listed in the manifest.

7. After Phase 1, which phase follows?

A. Phase 2

B. Phase 0

C. Phase 3

D. Phase 1

Phases are arranged in numerical order, signaling a forward progression through the steps. The next phase after Phase 1 is Phase 2 because you move to the immediately following number to continue the sequence. This reflects the built-in assumption that each phase advances the process rather than looping or jumping ahead; going to Phase 0 would precede Phase 1, and jumping straight to Phase 3 would skip Phase 2.

8. What does git filter-repo do?

A. It converts repository formats.

B. It creates backups automatically.

C. It removes passwords or renaming of specific files or file sets.

D. It integrates with issue trackers.

git filter-repo is a tool for rewriting history in a Git repository. It lets you surgically modify commits and the repository's trees, which is especially useful for scrubbing sensitive data or reorganizing the project layout. You can remove passwords, secrets, or entire sets of files from past commits, and you can rename or relocate files or directories across the entire history. This capability is what makes it the go-to solution for cleaning up a repo after secrets were committed or when you need to change how certain files are stored, without changing the current working state. It's faster and more flexible than older methods for history rewrites, and it isn't primarily designed for converting repository formats, automating backups, or integrating with issue trackers, which is why those options don't fit as well.

9. How many languages does CodeQL support in GAS and how do you enable them?

A. Multiple languages (e.g., Java, JavaScript/TypeScript, Python, C/C++, Go); configure the codeql-analysis workflow to enable relevant languages

B. Only Java and JavaScript

C. It supports only one language per workflow

D. It cannot be configured; it auto-detects languages

CodeQL in GAS can analyze multiple languages in a single analysis. You enable them by configuring the codeql-analysis workflow and listing the languages you want to analyze in the languages input of the action. For example, you can include languages like Java, JavaScript/TypeScript, Python, C/C++, and Go. This tells CodeQL to build databases for the selected languages and run the queries against them in one workflow run, so you're not limited to a single language.

10. What is the process for triaging a Code Scanning alert?

A. Open the alert, review location and rule, determine severity, reproduce if needed, assign to a developer, and decide on remediation or dismissal.

B. Wait for automatic escalation by system.

C. Delete the alert and ignore.

D. Submit a pull request to remove the rule.

Triaging a Code Scanning alert is about quickly validating and prioritizing findings by looking at exactly where in the code the issue was detected and which rule fired. That context helps you understand what kind of weakness or vulnerability is being flagged and how risky it might be. Next, you determine the severity to gauge urgency and follow-up effort. If the finding isn't immediately clear, you reproduce the alert or verify it to avoid chasing a false positive. Then you assign the alert to the developer or team responsible for that code area so the right person handles it. Finally, you decide on remediation or dismissal: fix the code if needed, apply an appropriate suppression or exception with clear notes if it's not a real issue, or mark it as dismissed if it's a known false positive. This sequence keeps alerts actionable and prevents them from being ignored or mishandled.

Next Steps

Congratulations on reaching the final section of this guide. You've taken a meaningful step toward passing your certification exam and advancing your career.

As you continue preparing, remember that consistent practice, review, and self-reflection are key to success. Make time to revisit difficult topics, simulate exam conditions, and track your progress along the way.

If you need help, have suggestions, or want to share feedback, we'd love to hear from you. Reach out to our team at hello@examzify.com.

Or visit your dedicated course page for more study tools and resources:

https://githubadvsec.examzify.com

We wish you the very best on your exam journey. You've got this!