DUMPSBOSS.

GitHub Advanced Security GHAS Exam

GitHub GitHub-Advanced-Security

Version Demo

Total Demo Questions: 9

Total Premium Questions: 95

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Describe GitHub Advanced Security	6
Topic 2, Configure and use secret scanning	23
Topic 3, Configure and use code scanning	32
Topic 4, Configure and use dependency management	31
Topic 5, Mix Questions	3
Total	95

QUESTION NO: 1

A CodeQL workflow uses custom build commands for a Java application. Which ordering requirements must the workflow meet for CodeQL to analyze the compiled code? (Each answer presents part of the solution. Choose two.)

A. Initialize CodeQL before running the custom build commands.

B. Run the analysis after the build commands complete.

C. Run the custom build commands before CodeQL initialization.

D. Upload the SARIF results before running the analysis.

ANSWER: A B

Explanation:

Initialize CodeQL before running the custom build commands is required so that CodeQL can monitor compilation and extract the information needed for the CodeQL database. **Run the analysis after the build commands complete** is also required because analysis evaluates the database created from the extracted code.

Building before CodeQL initialization prevents CodeQL from observing the compilation. Uploading SARIF before analysis is impossible because the analysis is the step that produces the results to upload.

QUESTION NO: 2

Where can you use CodeQL analysis for code scanning? (Each answer presents part of the solution. Choose two.)

A. In a third-party Git repository

B. In a workflow

C. In an external continuous integration (CI) system

D. In the Files changed tab of the pull request

ANSWER: B C

Explanation:

CodeQL analysis can be run **in a workflow**, most commonly through a GitHub Actions workflow configured with the CodeQL action. The workflow defines the events that trigger scanning, checks out the repository, initializes CodeQL for the selected languages, builds the code when necessary, performs analysis, and uploads the resulting SARIF findings to GitHub code scanning. This integrates analysis directly into the repository's development and pull-request process.

CodeQL analysis can also be run **in an external continuous integration (CI) system**. Organizations that use a CI provider other than GitHub Actions can install and run the CodeQL CLI in that environment, then upload the generated SARIF results to GitHub. GitHub processes the upload and displays the resulting code-scanning alerts in the repository's Security views, allowing externally generated analysis to participate in the same alert-management experience.

GitHub documents both GitHub Actions and third-party CI systems as supported ways to configure CodeQL code scanning. See Configuring code scanning with CodeQL and Uploading a SARIF file to GitHub.

QUESTION NO: 3

When configuring code scanning with CodeQL, what are your options for specifying additional queries? (Each answer presents part of the solution. Choose two.)

A. Packs

B. github/codeql

C. Scope

D. Queries

ANSWER: A D

Explanation:

Packs and **Queries** are the supported ways to add CodeQL analysis beyond the default query suite. A CodeQL query pack is a versioned, reusable package of queries that can be referenced in the CodeQL configuration. This enables teams to consistently share and run a curated collection of checks across repositories. Packs can be specified by package name and, when appropriate, a version or compatible version range, allowing the configuration to use maintained query content.

Individual queries can also be specified directly. The CodeQL configuration supports references to query files, directories, query suites, or query pack contents through the `queries` setting. This is useful when an organization needs to add a particular custom `.ql` query, a directory of internally developed queries, or an official CodeQL query suite in addition to the default security queries. GitHub documents that the `queries` setting accepts both query packs and paths to queries or query suites, making these two concepts the relevant parts of the configuration solution. See GitHub's CodeQL advanced setup documentation and its guidance on CodeQL configuration files.

QUESTION NO: 4

If default code security settings have not been changed at the repository, organization, or enterprise level, which repositories receive Dependabot alerts?

A. Repositories owned by an enterprise account

B. Private repositories

C. None

D. Repositories owned by an organization

E. Public repositories

ANSWER: E

Explanation:

Public repositories receive Dependabot alerts under GitHub's default configuration. Dependabot continuously checks the dependency manifests and lock files in a repository against GitHub's advisory data. When GitHub identifies a vulnerable dependency, it creates a Dependabot alert in the repository's Security tab and can notify relevant users according to their notification settings.

This default is designed to help maintainers of open-source code identify known dependency vulnerabilities without requiring an initial repository-level setup step. The wording about settings not having been changed at repository, organization, or enterprise scope matters because those scopes can be used to manage security features centrally and may alter the effective configuration. With no such override, the documented default for Dependabot alerts applies to public repositories.

For more detail, see GitHub's About Dependabot alerts documentation, which describes the feature and its default availability, and GitHub's Configuring Dependabot alerts guidance for enabling and managing alerts.

QUESTION NO: 5

What does a CodeQL database of your repository contain?

A. A build for Go projects to set up the project

B. A build of the code and extracted data

C. Build commands for C/C++, C#, and Java

D. GitHub

ANSWER: B

Explanation:

A CodeQL database contains a relational representation of the repository's source code: for compiled languages, this is created from information captured while the code is built; for interpreted languages, it is created directly from the source. The extraction process records the code's structure, such as files, functions, classes, expressions, control flow, and data relationships, as queryable data. CodeQL then runs security and quality queries against that extracted data rather than simply scanning raw text. This allows queries to model how values and behavior move through the application and identify patterns associated with vulnerabilities or coding errors. Therefore, "A build of the code and extracted data" accurately describes the database's contents. GitHub's CodeQL documentation explains that database creation involves extracting a representation of the codebase and, where required, monitoring the build process to capture the information needed for analysis. See Configuring CodeQL code scanning in your workflow and About code scanning with CodeQL.

QUESTION NO: 6

Which key is required in the update settings of the Dependabot configuration file?

A. rebase-strategy

B. commit-message

C. assignees

D. package-ecosystem

ANSWER: D

Explanation:

`package-ecosystem` is required in each entry under the `updates` section of `.github/dependabot.yml`. This key tells Dependabot which supported dependency-management ecosystem it must inspect for that particular update configuration. For example, valid values include `npm`, `pip`, `maven`, `bundler`, and `github-actions`. Dependabot uses the selected ecosystem to identify the relevant manifest and lock files, determine available dependency updates, and create pull requests using the correct package-manager behavior.

An update configuration represents a specific dependency-update job, so its ecosystem must be explicitly declared rather than inferred. GitHub's Dependabot options reference lists `package-ecosystem` as a required key for every `updates` configuration, alongside other required configuration information such as the target directory and update schedule. The value must exactly match one of the ecosystems GitHub supports. This makes `package-ecosystem` the required key among the choices presented. For the authoritative schema, supported values, and configuration examples, see GitHub's Dependabot options reference and configuration guide for Dependabot version updates.

QUESTION NO: 7

A repository uses a dependency review workflow to prevent risky dependency changes from being merged. Which policy types can the Dependency Review Action enforce when configured? (Each answer presents a complete solution. Choose two.)

A. A minimum vulnerability severity that causes the check to fail.

B. A list of disallowed licenses.

C. A requirement that all detected secrets have validity checks.

D. A required CodeQL query suite for every pull request.

E. A maximum number of open secret scanning alerts.

ANSWER: A B

Explanation:

A minimum vulnerability severity that causes the check to fail can be enforced with the action's severity configuration. This allows a repository to block pull requests that introduce dependencies with vulnerabilities at or above the selected severity.

A list of disallowed licenses can also be enforced. The action can fail when a newly introduced dependency has a license that the organization does not permit. Secret validity and CodeQL query suites are controlled by separate GitHub security features and are not dependency review policies.

QUESTION NO: 8

A build system resolves dependencies that are generated at build time and are not represented in any supported manifest or lock file committed to the repository. How should the build system make these dependencies available to GitHub security features?

A. Submit a dependency snapshot through the dependency submission API.

B. Upload the resolved dependencies as a SARIF file.

C. Create a CodeQL database for the build output.

D. Publish a repository security advisory for the dependencies.

ANSWER: A

Explanation:

Submit a dependency snapshot through the dependency submission API is correct because the API allows external build tooling to submit the resolved dependency information to the repository's dependency graph. Dependabot alerts and dependency review can then use that dependency information where applicable.

Uploading a SARIF file reports static-analysis findings, not dependency inventory. Creating a CodeQL database or a repository security advisory likewise does not add generated dependencies to the dependency graph.

QUESTION NO: 9

If notification and alert recipients are not customized, which users receive notifications about new Dependabot alerts in an affected repository?

A. Users with Write permissions to the repository

B. Users with Admin privileges to the repository

C. Users with Maintain privileges to the repository

D. Users with Read permissions to the repository

E. Users with Write, Maintain, or Admin permissions to the repository

ANSWER: E

Explanation:

Users with Write, Maintain, or Admin permissions to the repository receive notifications by default when GitHub creates a new Dependabot alert for that repository. This default recipient group ensures that people who can directly contribute to, manage, or administer the repository are informed about vulnerable dependencies and can take appropriate remediation action. The relevant access levels include the standard Write role as well as the higher-privilege Maintain and Admin roles; therefore, the complete group must be identified rather than naming only one role. Repository owners can customize

Dependabot alert notification recipients when their workflow requires a different distribution of security information, such as assigning alert ownership to a security team. Until that customization is made, GitHub applies the default notification behavior to the full set of users holding these repository permissions. Dependabot alerts identify known vulnerabilities in dependencies detected from repository manifests and lock files, helping authorized maintainers prioritize upgrades, removals, or other mitigations. GitHub documents the default recipients and available notification configuration in its Dependabot alert notification documentation. For the repository role definitions underlying these permissions, see Repository roles for an organization.