ATMIC NETWORKS
YOUR GUIDE TO GLOBAL CERTIFIATION

"Please note that these files may not be up to date. However, the questions will help you understand the exam format and typical question patterns."

www.atmicnetworks .com

Warning: Keep connected with our support team for latest updates

Question: 1

– [Configure and Use Code Scanning]

After investigating a code scanning alert related to injection, you determine that the input is properly sanitized using custom logic. What should be your next step?

A. Draft a pull request to update the open-source query.

B. Ignore the alert.

C. Open an issue in the CodeQL repository.

D. Dismiss the alert with the reason "false positive."

Answer: D

Explanation:

When you identify that a code scanning alert is a false positive—such as when your code uses a custom sanitization method not recognized by the analysis—you should dismiss the alert with the reason "false positive." This action helps improve the accuracy of future analyses and maintains the relevance of your security alerts.

As per GitHub's documentation:

"If you dismiss a CodeQL alert as a false positive result, for example because the code uses a sanitization library that isn't supported, consider contributing to the CodeQL repository and improving the analysis."

By dismissing the alert appropriately, you ensure that your codebase's security alerts remain actionable and relevant.

Question: 2

– [Configure and Use Dependency Management]

When does Dependabot alert you of a vulnerability in your software development process?

A. When a pull request adding a vulnerable dependency is opened

B. As soon as a vulnerable dependency is detected

C. As soon as a pull request is opened by a contributor

D. When Dependabot opens a pull request to update a vulnerable dependency

Answer: B

Explanation:

Dependabot alerts are generated as soon as GitHub detects a known vulnerability in one of your dependencies. GitHub does this by analyzing your repository's dependency graph and matching it against vulnerabilities listed in the GitHub Advisory Database. Once a match is found, the system raises an alert automatically without waiting for a PR or manual action.

This allows organizations to proactively mitigate vulnerabilities as early as possible, based on real-time detection.

Reference: GitHub Docs – About Dependabot alerts; Managing alerts in GitHub Dependabot

Question: 3

– [Configure and Use Dependency Management]

Which of the following is the most complete method for Dependabot to find vulnerabilities in third-party dependencies?

A. Dependabot reviews manifest files in the repository

B. CodeQL analyzes the code and raises vulnerabilities in third-party dependencies

C. A dependency graph is created, and Dependabot compares the graph to the GitHub Advisory database

D. The build tool finds the vulnerable dependencies and calls the Dependabot API

Answer: C

Explanation:

Dependabot builds a dependency graph by analyzing package manifests and lockfiles in your repository. This graph includes both direct and transitive dependencies. It then compares this graph against the GitHub Advisory Database, which includes curated, security-reviewed advisories.

This method provides a comprehensive and automated way to discover all known vulnerabilities across your dependency tree.

Reference: GitHub Docs – About the dependency graph; About Dependabot alerts

Question: 4

– [Describe the GHAS Security Features and Functionality]

What is a security policy?

A. An automatic detection of security vulnerabilities and coding errors in new or modified code

B. A security alert issued to a community in response to a vulnerability

C. A file in a GitHub repository that provides instructions to users about how to report a security vulnerability

D. An alert about dependencies that are known to contain security vulnerabilities

Answer: C

Explanation:

A security policy is defined by a SECURITY.md file in the root of your repository or .github/ directory. This file informs contributors and security researchers about how to responsibly report vulnerabilities. It improves your project's transparency and ensures timely communication and mitigation of any reported issues.

Adding this file also enables a "Report a vulnerability" button in the repository's Security tab.

Reference: GitHub Docs – Adding a security policy to your repository

Question: 5

– [Configure GitHub Advanced Security Tools in GitHub Enterprise]

As a repository owner, you want to receive specific notifications, including security alerts, for an individual repository. Which repository notification setting should you use?

A. Ignore

B. Participating and @mentions

C. All Activity

D. Custom

Answer: D

Explanation:

Using the Custom setting allows you to subscribe to specific event types, such as Dependabot alerts or vulnerability notifications, without being overwhelmed by all repository activity. This is essential for repository maintainers who need fine-grained control over what kinds of events trigger notifications.

This setting is configurable per repository and allows users to stay aware of critical issues while minimizing notification noise.

Reference: GitHub Docs – Configuring notifications; Managing security alerts

Question: 6

– [Configure GitHub Advanced Security Tools in GitHub Enterprise]

Which of the following Watch settings could you use to get Dependabot alert notifications? (Each answer presents part of the solution. Choose two.)

A. The Custom setting

B. The Participating and @mentions setting

C. The All Activity setting

D. The Ignore setting

Answer: A, C

Explanation:

Comprehensive and Detailed Explanation:

To receive Dependabot alert notifications for a repository, you can utilize the following Watch settings:

Custom setting: Allows you to tailor your notifications, enabling you to subscribe specifically to security alerts, including those from Dependabot.

All Activity setting: Subscribes you to all notifications for the repository, encompassing issues, pull requests, and security alerts like those from Dependabot.

The Participating and @mentions setting limits notifications to conversations you're directly involved in or mentioned, which may not include security alerts. The Ignore setting unsubscribes you from all notifications, including critical security alerts.

GitHub Docs

+1

GitHub Docs

+1

Reference: GitHub Docs – Configuring notifications; Managing security alerts

Question: 7

– [Configure and Use Dependency Management]

Which Dependabot configuration fields are required? (Each answer presents part of the solution. Choose three.)

A. directory
B. package-ecosystem
C. milestone
D. schedule.interval
E. allow

Answer: A, B, D

Explanation:

Comprehensive and Detailed Explanation:

When configuring Dependabot via the dependabot.yml file, the following fields are mandatory for each update configuration:

directory: Specifies the location of the package manifest within the repository. This tells Dependabot where to look for dependency files.

package-ecosystem: Indicates the type of package manager (e.g., npm, pip, maven) used in the specified directory.

schedule.interval: Defines how frequently Dependabot checks for updates (e.g., daily, weekly). This ensures regular scanning for outdated or vulnerable dependencies.

The milestone field is optional and used for associating pull requests with milestones. The allow field is also optional and used to specify which dependencies to update.

GitLab

Reference: GitHub Docs – Configuration options for dependency updates

Question: 8

– [Configure and Use Code Scanning]

What is required to trigger code scanning on a specified branch?

A. The repository must be private.

B. Secret scanning must be enabled on the repository.

C. Developers must actively maintain the repository.

D. The workflow file must exist in that branch.

Answer: D

Explanation:

Comprehensive and Detailed Explanation:

For code scanning to be triggered on a specific branch, the branch must contain the appropriate workflow file, typically located in the .github/workflows directory. This YAML file defines the code scanning configuration and specifies the events that trigger the scan (e.g., push, pull_request).

Without the workflow file in the branch, GitHub Actions will not execute the code scanning process for that branch. The repository's visibility (private or public), the status of secret scanning, or the activity level of developers do not directly influence the triggering of code scanning.

Reference: GitHub Docs – About workflows; About code scanning alerts

Question: 9

– [Describe GitHub Advanced Security Best Practices]

As a contributor, you discovered a vulnerability in a repository. Where should you look for the instructions on how to report the vulnerability?

A. support.md

B. readme.md

C. contributing.md

D. security.md

Answer: D

Explanation:

The correct place to look is the SECURITY.md file. This file provides contributors and security researchers with instructions on how to responsibly report vulnerabilities. It may include contact methods, preferred communication channels (e.g., security team email), and disclosure guidelines.

This file is considered a GitHub best practice and, when present, activates a "Report a vulnerability" button in the repository's Security tab.

Reference: GitHub Docs – Adding a security policy to your repository

Question: 10

– [Configure and Use Dependency Management]

Assuming there is no custom Dependabot behavior configured, where possible, what does Dependabot do after sending an alert about a vulnerable dependency in a repository?

A. Creates a pull request to upgrade the vulnerable dependency to the minimum possible secure version

B. Scans repositories for vulnerable dependencies on a schedule and adds those files to a manifest

C. Constructs a graph of all the repository's dependencies and public dependents for the default branch

D. Scans any push to all branches and generates an alert for each vulnerable repository

Answer: A

Explanation:

After generating an alert for a vulnerable dependency, Dependabot automatically attempts to create a pull request to upgrade that dependency to the minimum required secure version—if a fix is available and compatible with your project.

This automated PR helps teams fix vulnerabilities quickly with minimal manual intervention. You can also configure update behaviors using dependabot.yml, but in the default state, PR creation is automatic.

Reference: GitHub Docs – About Dependabot alerts; About Dependabot security updates

Question: 11

– [Configure and Use Secret Scanning]

What is the first step you should take to fix an alert in secret scanning?

A. Archive the repository.

B. Update your dependencies.

C. Revoke the alert if the secret is still valid.

D. Remove the secret in a commit to the main branch.

Answer: C

Explanation:

The first step when you receive a secret scanning alert is to revoke the secret if it is still valid. This ensures the secret can no longer be used maliciously. Only after revoking it should you proceed to remove it from the code history and apply other mitigation steps.

Simply deleting the secret from the code does not remove the risk if it hasn't been revoked — especially since it may already be exposed in commit history.

Reference: GitHub Docs – About secret scanning alerts; Remediating a secret scanning alert

Question: 12

– [Configure and Use Dependency Management]
A dependency has a known vulnerability. What does the warning message include?

A. The security impact of these changes

B. An easily understandable visualization of dependency change

C. How many projects use these components

D. A brief description of the vulnerability

Answer: D

Explanation:

When a vulnerability is detected, GitHub shows a warning that includes a brief description of the vulnerability. This typically covers the name of the CVE (if available), a short summary of the issue, severity level, and potential impact. The message also links to additional advisory data from the GitHub Advisory Database.

This helps developers understand the context and urgency of the vulnerability before applying the fix.

Reference: GitHub Docs – About Dependabot alerts; Reviewing and managing alerts

Question: 13

– [Configure and Use Dependency Management]

Assuming that notification and alert recipients are not customized, what does GitHub do when it identifies a vulnerable dependency in a repository where Dependabot alerts are enabled? (Each answer presents part of the solution. Choose two.)

A. It generates a Dependabot alert and displays it on the Security tab for the repository.
B. It notifies the repository administrators about the new alert.

C. It generates Dependabot alerts by default for all private repositories.

D. It consults with a security service and conducts a thorough vulnerability review.

Answer: A, B

Explanation:

Comprehensive and Detailed Explanation:

When GitHub identifies a vulnerable dependency in a repository with Dependabot alerts enabled, it performs the following actions:

Generates a Dependabot alert: The alert is displayed on the repository's Security tab, providing details about the vulnerability and affected dependency.

Notifies repository maintainers: By default, GitHub notifies users with write, maintain, or admin permissions about new Dependabot alerts.

GitHub Docs

These actions ensure that responsible parties are informed promptly to address the vulnerability.

Reference: GitHub Docs – About Dependabot alerts; Configuring notifications for Dependabot alerts

Question: 14

– [Configure and Use Secret Scanning]

What do you need to do before you can define a custom pattern for a repository?

A. Provide a regular expression for the format of your secret pattern.

B. Add a secret scanning custom pattern.

C. Enable secret scanning on the repository.

D. Provide match requirements for the secret format.

Stack Overflow

Answer: C

Explanation:

Comprehensive and Detailed Explanation:

Before defining a custom pattern for secret scanning in a repository, you must enable secret scanning for that repository. Secret scanning must be active to utilize custom patterns, which allow you to define specific formats (using regular expressions) for secrets unique to your organization.
Once secret scanning is enabled, you can add custom patterns to detect and prevent the exposure of sensitive information tailored to your needs.

Reference: GitHub Docs – Managing alerts from secret scanning

Question: 15

– [Configure and Use Dependency Management]

Assuming that no custom Dependabot behavior is configured, who has the ability to merge a pull request created via Dependabot security updates?

A. An enterprise administrator

B. A user who has write access to the repository

C. A user who has read access to the repository

D. A repository member of an enterprise organization

Answer: B

Explanation:

Comprehensive and Detailed Explanation:
By default, users with write access to a repository have the ability to merge pull requests, including those created by Dependabot for security updates. This access level allows contributors to manage and integrate changes, ensuring that vulnerabilities are addressed promptly.

Users with only read access cannot merge pull requests, and enterprise administrators do not automatically have merge rights unless they have write or higher permissions on the specific repository.

Reference: GitHub Docs – About Dependabot security updates; Configuring Dependabot security updates

Question: 16

– [Configure and Use Code Scanning]

Who can fix a code scanning alert on a private repository?

A. Users who have the Triage role within the repository

B. Users who have Read permissions within the repository

C. Users who have Write access to the repository

D. Users who have the security manager role within the repository

Answer: C

Explanation:
Comprehensive and Detailed Explanation:

In private repositories, users with write access can fix code scanning alerts. They can do this by committing changes that address the issues identified by the code scanning tools. This level of access ensures that only trusted contributors can modify the code to resolve potential security vulnerabilities.

GitHub Docs

Users with read or triage roles do not have the necessary permissions to make code changes, and the security manager role is primarily focused on managing security settings rather than directly modifying code.

Reference: GitHub Docs – Resolving code scanning alerts

GitHub Docs

Question: 17

– [Describe the GHAS Security Features and Functionality]

Which of the following information can be found in a repository's Security tab?

A. Number of alerts per GHAS feature

B. Two-factor authentication (2FA) options

C. Access management

D. GHAS settings

Answer: A

Explanation:

The Security tab in a GitHub repository provides a central location for viewing security-related information, especially when GitHub Advanced Security is enabled. The following can be accessed:

Number of alerts related to:

Code scanning

Secret scanning

Dependency (Dependabot) alerts

Summary and visibility into open, closed, and dismissed security issues.

It does not show 2FA options, access control settings, or configuration panels for GHAS itself. Those belong to account or organization-level settings.

Reference: GitHub Docs – Managing security and analysis settings for your repository

Question: 18

– [Configure and Use Secret Scanning]

How many alerts are created when two instances of the same secret value are in the same repository?

A. 1

B. 2

C. 3

D. 4

Answer: A

Explanation:

When multiple instances of the same secret value appear in a repository, only one alert is generated. Secret scanning works by identifying exposed credentials and token patterns, and it groups identical matches into a single alert to reduce noise and avoid duplication.

This makes triaging easier and helps teams focus on remediating the actual exposed credential rather than reviewing multiple redundant alerts.

Reference: GitHub Docs – About secret scanning alerts

Question: 19

– [Configure and Use Secret Scanning]

What happens when you enable secret scanning on a private repository?

A. Repository administrators can view Dependabot alerts.

B. Your team is subscribed to security alerts.

C. GitHub performs a read-only analysis on the repository.

D. Dependency review, secret scanning, and code scanning are enabled.

Answer: C

Explanation:

When secret scanning is enabled on a private repository, GitHub performs a read-only analysis of the repository's contents. This includes the entire Git history and files to identify strings that match known secret patterns or custom-defined patterns.

GitHub does not alter the repository, and enabling secret scanning does not automatically enable code scanning or dependency review — each must be configured separately.

Reference: GitHub Docs – Managing secret scanning for repositories

Question: 20

– [Configure and Use Dependency Management]

You have enabled security updates for a repository. When does GitHub mark a Dependabot alert as resolved for that repository?

A. When Dependabot creates a pull request to update dependencies

B. When you dismiss the Dependabot alert

C. When the pull request checks are successful

D. When you merge a pull request that contains a security update

Answer: D

Explanation:

A Dependabot alert is marked as resolved only after the related pull request is merged into the repository. This indicates that the vulnerable dependency has been officially replaced with a secure version in the active codebase.

Simply generating a PR or passing checks does not change the alert status; merging is the key step.

Reference: GitHub Docs – About Dependabot security updates; Managing Dependabot alerts

Question: 21

– [Use Code Scanning with CodeQL]

How would you build your code within the CodeQL analysis workflow? (Each answer presents a complete solution. Choose two.)

A. Upload compiled binaries.

B. Use CodeQL's init action.

C. Ignore paths.

D. Implement custom build steps.

E. Use jobs.analyze.runs-on.

F. Use CodeQL's autobuild action.

Answer: D, F

Explanation:

Comprehensive and Detailed Explanation:

When setting up CodeQL analysis for compiled languages, there are two primary methods to build your code:

GitHub Docs

Autobuild: CodeQL attempts to automatically build your codebase using the most likely build method. This is suitable for standard build processes.
GitHub Docs

Custom Build Steps: For complex or non-standard build processes, you can implement custom build steps by specifying explicit build commands in your workflow. This provides greater control over the build process.

GitHub Docs

The init action initializes the CodeQL analysis but does not build the code. The jobs.analyze.runs-on specifies the operating system for the runner but is not directly related to building the code. Uploading compiled binaries is not a method supported by CodeQL for analysis.

Reference: GitHub Docs – CodeQL code scanning for compiled languages

Question: 22

– [Configure and Use Dependency Management]

Which of the following workflow events would trigger a dependency review? (Each answer presents a complete solution. Choose two.)

A. pull_request

B. workflow_dispatch

C. trigger

D. commit

Answer: A, B

Explanation:

Comprehensive and Detailed Explanation:

Dependency review is triggered by specific events in GitHub workflows:

pull_request: When a pull request is opened, synchronized, or reopened, GitHub can analyze the changes in dependencies and provide a dependency review.

workflow_dispatch: This manual trigger allows users to initiate workflows, including those that perform dependency reviews.

The trigger and commit options are not recognized GitHub Actions events and would not initiate a dependency review.

Reference: GitHub Docs – Events that trigger workflows

Question: 23

– [Configure and Use Dependency Management]

You are a maintainer of a repository and Dependabot notifies you of a vulnerability. Where could the vulnerability have been disclosed? (Each answer presents part of the solution. Choose two.)

A. In the National Vulnerability Database
B. In the dependency graph
C. In security advisories reported on GitHub
D. In manifest and lock files

Answer: A, C

Explanation:

Comprehensive and Detailed Explanation:

Dependabot alerts are generated based on data from various sources:

National Vulnerability Database (NVD): A comprehensive repository of known vulnerabilities, which GitHub integrates into its advisory database.

GitHub Docs

Security Advisories Reported on GitHub: GitHub allows maintainers and security researchers to report and discuss vulnerabilities, which are then included in the advisory database.

The dependency graph and manifest/lock files are tools used by GitHub to determine which dependencies are present in a repository but are not sources of vulnerability disclosures themselves.

Reference: GitHub Docs – About Dependabot alerts

Question: 24

– [Configure and Use Secret Scanning]

Which of the following statements most accurately describes push protection for secret scanning custom patterns?

A. Push protection must be enabled for all, or none, of a repository's custom patterns.
B. Push protection is an opt-in experience for each custom pattern.
C. Push protection is not available for custom patterns.
D. Push protection is enabled by default for new custom patterns.

Answer: B

Explanation:

Comprehensive and Detailed Explanation:

Push protection for secret scanning custom patterns is an opt-in feature. This means that for each custom pattern defined in a repository, maintainers can choose to enable or disable push protection individually. This provides flexibility, allowing teams to enforce push protection on sensitive patterns while leaving it disabled for others.

Reference: GitHub Docs – Working with push protection from the command line

Question: 25

– [Use Code Scanning with CodeQL]

When using the advanced CodeQL code scanning setup, what is the name of the workflow file?

A. codeql-config.yml
B. codeql-scan.yml
C. codeql-workflow.yml
D. codeql-analysis.yml

Answer: D

Explanation:

Comprehensive and Detailed Explanation:

In the advanced setup for CodeQL code scanning, GitHub generates a workflow file named codeql-analysis.yml. This file is located in the .github/workflows directory of your repository. It defines the configuration for the CodeQL analysis, including the languages to analyze, the events that trigger the analysis, and the steps to perform during the workflow.
Reference: GitHub Docs – Customizing your advanced setup for code scanning

Question: 26

– [Configure and Use Secret Scanning]

Which of the following statements best describes secret scanning push protection?

A. Commits that contain secrets are blocked before code is added to the repository.

B. Secret scanning alerts must be closed before a branch can be merged into the repository.

C. Buttons for sensitive actions in the GitHub UI are disabled.

D. Users need to reply to a 2FA challenge before any push events.

Answer: A

Explanation:

Comprehensive and Detailed Explanation:

Secret scanning push protection is a proactive feature that scans for secrets in your code during the push process. If a secret is detected, the push is blocked, preventing the secret from being added to the repository. This helps prevent accidental exposure of sensitive information.
GitHub Docs

Reference: GitHub Docs – About push protection

Question: 27

– [Configure and Use Dependency Management]

In a private repository, what minimum requirements does GitHub need to generate a dependency graph? (Each answer presents part of the solution. Choose two.)

A. Read-only access to all the repository's files
B. Dependency graph enabled at the organization level for all new private repositories
C. Write access to the dependency manifest and lock files for an enterprise
D. Read-only access to the dependency manifest and lock files for a repository

Answer: B, D

Explanation:
Comprehensive and Detailed Explanation:

To generate a dependency graph for a private repository, GitHub requires:

Dependency graph enabled: The repository must have the dependency graph feature enabled. This can be configured at the organization level to apply to all new private repositories.

Access to manifest and lock files: GitHub needs read-only access to the repository's dependency manifest and lock files (e.g., package.json, requirements.txt) to identify and map dependencies.

Reference: GitHub Docs – About the dependency graph

Question: 28

– [Use Code Scanning with CodeQL]

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. A representation of all of the source code

GitHub

Agentic AI for AppSec Teams

Answer: B

Explanation:

Comprehensive and Detailed Explanation:

A CodeQL database contains a representation of your codebase, including the build of the code and extracted data. This database is used to run CodeQL queries to analyze your code for potential vulnerabilities and errors.

GitHub Docs

Reference: GitHub Docs – Preparing your code for CodeQL analysis

Question: 29

– [Use Code Scanning with CodeQL]

When using CodeQL, how does extraction for compiled languages work?

A. By generating one language at a time
B. By resolving dependencies to give an accurate representation of the codebase
C. By monitoring the normal build process
D. By running directly on the source code

Answer: C

Explanation:

For compiled languages, CodeQL performs extraction by monitoring the normal build process. This means it watches your usual build commands (like make, javac, or dotnet build) and extracts the relevant data from the actual build steps being executed. CodeQL uses this information to construct a semantic database of the application.

This approach ensures that CodeQL captures a precise, real-world representation of the code and its behavior as it is compiled, including platform-specific configurations or conditional logic used during build.

Reference: GitHub Docs – CodeQL for compiled languages

Question: 30

– [Configure and Use Secret Scanning]

Which of the following features helps to prioritize secret scanning alerts that present an immediate risk?

A. Non-provider patterns
B. Push protection
C. Custom pattern dry runs
D. Secret validation

Answer: D

Explanation:

Secret validation checks whether a secret found in your repository is still valid and active with the issuing provider (e.g., AWS, GitHub, Stripe). If a secret is confirmed to be active, the alert is marked as verified, which means it's considered a high-priority issue because it presents an immediate security risk.

This helps teams respond faster to valid, exploitable secrets rather than wasting time on expired or fake tokens.

Reference: GitHub Docs – Secret validation in secret scanning

Question: 31

– [Configure and Use Secret Scanning]

What is a prerequisite to define a custom pattern for a repository?

A. Change the repository visibility to Internal
B. Close other secret scanning alerts
C. Specify additional match criteria
D. Enable secret scanning

Answer: D

Explanation:

You must enable secret scanning before defining custom patterns. Secret scanning provides the foundational capability for detecting exposed credentials, and custom patterns build upon that by allowing organizations to specify their own regex-based patterns for secrets unique to their environment.

Without enabling secret scanning, GitHub will not process or apply custom patterns.

Reference: GitHub Docs – Custom patterns in secret scanning

Question: 32

– [Use Code Scanning with CodeQL]

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:

In a workflow: GitHub Actions workflows are the most common place for CodeQL code scanning. The codeql-analysis.yml defines how the analysis runs and when it triggers.

In an external CI system: GitHub allows you to run CodeQL analysis outside of GitHub Actions. Once complete, the results can be uploaded using the upload-sarif action to make alerts visible in the repository.

You cannot run or trigger analysis from third-party repositories directly, and the Files changed tab in pull requests only shows diff — not analysis results.

Reference: GitHub Docs – Using CodeQL with CI and workflows

Question: 33

– [Use Code Scanning with CodeQL]

Where can you view code scanning results from CodeQL analysis?

A. The repository's code scanning alerts
B. A CodeQL database
C. A CodeQL query pack
D. At Security advisories

Answer: A

Explanation:

All results from CodeQL analysis appear under the repository's code scanning alerts tab. This section is part of the Security tab and provides a list of all current, fixed, and dismissed alerts found by CodeQL.

A CodeQL database is used internally during scanning but does not display results. Query packs contain rules, not results. Security advisories are for published vulnerabilities, not per-repo findings.

Reference: GitHub Docs – Viewing code scanning alerts

Question: 34

– [Use Code Scanning with CodeQL]

When using CodeQL, what extension stores query suite definitions?

A. .yml

B. .ql

C. .qll

D. .qls

Answer: D

Explanation:

Query suite definitions in CodeQL are stored using the .qls file extension. A query suite defines a collection of queries to be run during an analysis and allows for grouping them based on categories like language, security relevance, or custom filters.

In contrast:

.ql files are individual queries.

.qll files are libraries used by .ql queries.

.yml is used for workflows, not query suites.

Reference: GitHub Docs – CodeQL Query Suite Files

Question: 35

– [Configure GitHub Advanced Security Tools in GitHub Enterprise]

What role is required to change a repository's code scanning severity threshold that fails a pull request status check?

A. Maintain

B. Write

C. Triage

D. Admin

Answer: D

Explanation:

To change the threshold that defines whether a pull request fails due to code scanning alerts (such as blocking merges based on severity), the user must have Admin access on the repository. This is because modifying these settings falls under repository configuration privileges.

Users with Write, Maintain, or Triage roles do not have the required access to modify rulesets or status check policies.

Reference: GitHub Docs – Repository Role Permissions

Question: 36

– [Use Code Scanning with CodeQL]

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: AD

Explanation:

You can customize CodeQL scanning by including additional query packs or by specifying individual queries:

Packs: These are reusable collections of CodeQL queries bundled into a single package.

Queries: You can point to specific files or directories containing .ql queries to include in the analysis.

github/codeql refers to a pack by name but is not a method or field. Scope is not a valid field used for configuration in this context.

Reference: GitHub Docs – Customizing CodeQL Workflows

Question: 37

– [Configure GitHub Advanced Security Tools in GitHub Enterprise]

What step is required to run a SARIF-compatible (Static Analysis Results Interchange Format) tool on GitHub Actions?

A. Update the workflow to include a final step that uploads the results.

B. By default, the CodeQL runner automatically uploads results to GitHub on completion.

C. The CodeQL action uploads the SARIF file automatically when it completes analysis.

D. Use the CLI to upload results to GitHub.

Answer: A

Explanation:

When using a SARIF-compatible tool within GitHub Actions, it's necessary to explicitly add a step in your workflow to upload the analysis results. This is typically done using the upload-sarif action, which takes the SARIF file generated by your tool and uploads it to GitHub for processing and display in the Security tab. Without this step, the results won't be available in GitHub's code scanning interface.

Reference: GitHub Docs – Uploading a SARIF file to GitHub

Question: 38

– [Assessing Code Scanning Alerts]

You are managing code scanning alerts for your repository. You receive an alert highlighting a problem with data flow. What do you click for additional context on the alert?

A. Show paths

B. Security

C. Code scanning alerts

Answer: A

Explanation:

When dealing with a data flow issue in a code scanning alert, clicking on "Show paths" provides a detailed view of the data's journey through the code. This includes the source of the data, the path it takes, and where it ends up (the sink).

This information is crucial for understanding how untrusted data might reach sensitive parts of your application and helps in identifying where to implement proper validation or sanitization.

Reference: GitHub Docs – Assessing code scanning alerts for your repositoryGitHub Docs+1GitHub Docs+1

Question: 39

– [Configure and Use Secret Scanning]

Secret scanning will scan:

A. A continuous integration system.
B. Any Git repository.
C. The GitHub repository.
D. External services.

Answer: C

Explanation:

Secret scanning is a feature provided by GitHub that scans the contents of your GitHub repositories for known types of secrets, such as API keys and tokens. It operates within the GitHub environment and does not scan external systems, services, or repositories outside of GitHub. Its primary function
is to prevent the accidental exposure of sensitive information within your GitHub-hosted code.

Reference: GitHub Docs – About secret scanning

Question: 40

– [Configure and Use Dependency Management]

Which of the following formats are used to describe a Dependabot alert? (Each answer presents a complete solution. Choose two.)

A. Common Weakness Enumeration (CWE)
B. Exploit Prediction Scoring System (EPSS)
C. Common Vulnerabilities and Exposures (CVE)
D. Vulnerability Exploitability exchange (VEX)

Answer: A, C

Explanation:

Dependabot alerts utilize standardized identifiers to describe vulnerabilities:

CVE (Common Vulnerabilities and Exposures): A widely recognized identifier for publicly known cybersecurity vulnerabilities.

CWE (Common Weakness Enumeration): A category system for software weaknesses and vulnerabilities.

These identifiers help developers understand the nature of the vulnerabilities and facilitate the search for more information or remediation strategies.

Reference: GitHub Docs – About Dependabot alerts

Question: 41

– [Describe GitHub Advanced Security Best Practices]

What kind of repository permissions do you need to request a Common Vulnerabilities and Exposures (CVE) identification number for a security advisory?

A. Maintain
B. Admin
C. Triage
D. Write

Answer: B

Explanation:

Requesting a CVE ID for a security advisory in a GitHub repository requires Admin permissions. This level of access is necessary because it involves managing sensitive security information and coordinating with external entities to assign a

CVE, which is a formal process that can impact the public perception and security posture of the project.

Reference: GitHub Docs – About repository security advisories

Question: 42

– [Use Code Scanning with CodeQL]

As a developer with write access, you navigate to a code scanning alert in your repository. When will GitHub close this alert?

A. After you triage the pull request containing the alert

B. When you use data-flow analysis to find potential security issues in code

C. After you find the code and click the alert within the pull request

D. After you fix the code by committing within the pull request

Answer: D

Explanation:

GitHub automatically closes a code scanning alert when the vulnerable code is fixed in the same branch where the alert was generated, usually via a commit inside a pull request. Simply clicking or triaging an alert does not resolve it. The alert is re-evaluated after each push to the branch, and if the issue no longer exists, it is marked as resolved.

Reference: GitHub Docs – Code Scanning Alerts Lifecycle

Question: 43

– [Configure and Use Secret Scanning]

Which details do you have to provide to create a custom pattern for secret scanning? (Each answer presents part of the solution. Choose two.)

A. The secret format

B. The name of the pattern

C. A list of repositories to scan

D. Additional match requirements for the secret format

Answer: AB

Explanation:
When defining a custom pattern for secret scanning, two key fields are required:

Name of the pattern: A unique label to identify the pattern

Secret format: A regular expression that defines what the secret looks like (e.g., token format)

You can optionally specify additional match requirements (like required context keywords), but they're not mandatory.

Listing repositories is also not part of the required fields during pattern creation.

Reference: GitHub Docs – Defining Custom Patterns for Secret Scanning

Question: 44

– [Configure and Use Dependency Management]

Assuming that notification settings and Dependabot alert recipients have not been customized, which user account setting should you use to get an alert when a vulnerability is detected in one of your repositories?

A. Enable all in existing repositories

B. Enable by default for new public repositories

C. Enable all for Dependabot alerts

D. Enable all for Dependency graph

Answer: C

Explanation:

To ensure you're notified whenever a vulnerability is detected via Dependabot, you must enable alerts for Dependabot in your personal notification settings. This applies to both new and existing repositories. It ensures you get timely alerts about security vulnerabilities.

The dependency graph must be enabled for scanning, but does not send alerts itself.

Reference: GitHub Docs – Configuring Notifications for Dependabot Alerts

Question: 45

– [Describe GitHub Advanced Security Best Practices]

Which of the following benefits do code scanning, secret scanning, and dependency review provide?

A. Search for potential security vulnerabilities, detect secrets, and show the full impact of changes to dependencies

B. Confidentially report security vulnerabilities and privately discuss and fix security vulnerabilities in your repository's code

C. View alerts about dependencies that are known to contain security vulnerabilities

D. Automatically raise pull requests, which reduces your exposure to older versions of dependencies

Answer: A

Explanation:

These three features provide a complete layer of defense:

Code scanning identifies security flaws in your source code

Secret scanning detects exposed credentials

Dependency review shows the impact of package changes during a pull request

Together, they give developers actionable insight into risk and coverage throughout the SDLC.

Reference: GitHub Docs – About GitHub Advanced Security Features

Question: 46

– [Configure and Use Secret Scanning]

Assuming security and analysis features are not configured at the repository, organization, or enterprise level, secret scanning is enabled on:

A. Public repositories
B. All new repositories within your organization
C. User-owned private repositories
D. Private repositories

Answer: A

Explanation:

By default, secret scanning is enabled automatically for all public repositories. For private or internal repositories, secret scanning must be enabled manually unless configured at the organization or enterprise level.

This default behavior helps protect open-source projects without requiring additional configuration.

Reference: GitHub Docs – Secret Scanning Defaults and Scope

Question: 47

E. [Configure and Use Secret Scanning]

When secret scanning detects a set of credentials on a public repository, what does GitHub do?

A. It notifies the service provider who issued the secret.
B. It displays a public alert in the Security tab of the repository.
C. It scans the contents of the commits for additional secrets.
D. It sends a notification to repository members.

Answer: A

Explanation:

When a public repository contains credentials that match known secret formats, GitHub will automatically notify the service provider that issued the secret. This process is known as "secret scanning partner notification". The provider may then revoke the secret or contact the user directly.

GitHub does not publicly display the alert and does not send internal repository notifications for public detections.

Reference: GitHub Docs – Secret Scanning for Public Repositories

Question: 48

– [Configure and Use Dependency Management]

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:

In a dependabot.yml configuration file, package-ecosystem is a required key. It defines the package manager being used in that update configuration (e.g., npm, pip, maven, etc.).

Without this key, Dependabot cannot determine how to analyze or update dependencies. Other keys like rebase-strategy or commit-message are optional and used for customizing behavior.

Reference: GitHub Docs – Dependabot Configuration Options

Question: 49

– [Describe GitHub Advanced Security Best Practices]

Which of the following tasks can be performed by a security team as a proactive measure to help address secret scanning

alerts? (Each answer presents a complete solution. Choose two.)

A. Dismiss alerts that are older than 90 days.

B. Configure a webhook to monitor for secret scanning alert events.

C. Enable system for cross-domain identity management (SCIM) provisioning for the enterprise.

D. Document alternatives to storing secrets in the source code.

Answer: BD

Explanation:

To proactively address secret scanning:

Webhooks can be configured to listen for secret scanning events. This allows automation, logging, or alerting in real-time when secrets are detected.

Documenting secure development practices (like using environment variables or secret managers) helps reduce the likelihood of developers committing secrets in the first place.

Dismissal based on age is not a best practice without triage. SCIM deals with user provisioning, not scanning alerts.

Reference: GitHub Docs – Managing and Responding to Secret Scanning Alerts

Question: 50

– [Configure and Use Secret Scanning]

What YAML syntax do you use to exclude certain files from secret scanning?

A. decrypt_secret.sh

B. paths-ignore:

C. branches-ignore:

D. secret scanning.yml

Answer: B

Explanation:

To exclude specific files or directories from being scanned by secret scanning in GitHub Actions, you can use the paths-ignore: key within your YAML workflow file.

This tells GitHub to ignore specified paths when scanning for secrets, which can be useful for excluding test data or non-sensitive mock content.

Other options listed are invalid:

branches-ignore: excludes branches, not files.

decrypt_secret.sh is not a YAML key.

secret scanning.yml is not a recognized filename for configuration.

Reference: GitHub Docs – Ignoring Files in GitHub Actions for Secret Scanning

Question: 51

– [Use Code Scanning with CodeQL]

Which of the following options are code scanning application programming interface (API) endpoints? (Each answer presents part of the solution. Choose two.)

A. List all open code scanning alerts for the default branch

B. Modify the severity of an open code scanning alert

C. Get a single code scanning alert

D. Delete all open code scanning alerts

Answer: AC

Explanation:

The GitHub Code Scanning API includes endpoints that allow you to:

List alerts for a repository (filtered by branch, state, or tool) — useful for monitoring security over time.

Get a single alert by its ID to inspect its metadata, status, and locations in the code.

However, GitHub does not support modifying the severity of alerts via API — severity is defined by the scanning tool (e.g., CodeQL). Likewise, alerts cannot be deleted via the API; they are resolved by fixing the code or dismissing them manually.

Reference: GitHub Docs – Code Scanning REST API

Question: 52

– [Use Code Scanning with CodeQL]

Why should you dismiss a code scanning alert?

A. If you fix the code that triggered the alert

B. To prevent developers from introducing new problems

C. If it includes an error in code that is used only for testing

D. If there is a production error in your code

Answer: C

Explanation:

You should dismiss a code scanning alert if the flagged code is not a true security concern, such as:

Code in test files

Code paths that are unreachable or safe by design

False positives from the scanner

Fixing the code would automatically resolve the alert — not dismiss it. Dismissing is for valid exceptions or noise reduction.

Reference: GitHub Docs – Dismissing Code Scanning Alerts

Question: 53

– [Configure and Use Dependency Management]

What should you do after receiving an alert about a dependency added in a pull request?

A. Disable Dependabot alerts for all repositories owned by your organization

B. Fork the branch and deploy the new fork

C. Update the vulnerable dependencies before the branch is merged

D. Deploy the code to your default branch

Answer: C

Explanation:

If an alert is raised on a pull request dependency, best practice is to update the dependency to a secure version before merging the PR. This prevents the vulnerable version from entering the main

codebase.

Merging or deploying the PR without fixing the issue exposes your production environment to known risks.

Reference: GitHub Docs – Reviewing Dependabot Alerts in Pull Requests

Question: 54

– [Configure and Use Secret Scanning]

Where can you find a deleted line of code that contained a secret value?

A. Insights

B. Issues

C. Commits

D. Dependency graph

Answer: C

Explanation:

Secrets committed and then deleted are still accessible in the repository's Git history. To locate them, navigate to the Commits tab. GitHub's secret scanning can detect secrets in both current and historical commits, which is why remediation should also include revoking the secret, not just removing it from the latest code.

Reference: GitHub Docs – Remediating Secret Scanning Alerts

Question: 55

– [Configure and Use Dependency Management]
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

Answer: C

Explanation:

By default, no repositories receive Dependabot alerts unless configuration is explicitly enabled. GitHub does not enable Dependabot alerts automatically for any repositories unless:

The feature is turned on manually

It's configured at the organization or enterprise level via security policies

This includes public, private, and enterprise-owned repositories — manual activation is required.

Reference: GitHub Docs – About Dependabot Alerts

Question: 56

– [Configure and Use Secret Scanning]

Which of the following secret scanning features can verify whether a secret is still active?

A. Push protection

B. Validity checks

C. Branch protection

D. Custom patterns

Answer: B

Explanation:

Validity checks, also called secret validation, allow GitHub to check if a detected secret is still active. If verified as live, the alert is marked as "valid", allowing security teams to prioritize the most critical leaks.

Push protection blocks secrets but does not check their validity. Custom patterns are user-defined and do not include live checks.

Reference: GitHub Docs – Secret Scanning Validity

Question: 57

– [Configure and Use Secret Scanning]

Which of the following is the best way to prevent developers from adding secrets to the repository?

A. Create a CODEOWNERS file

B. Make the repository public

C. Configure a security manager

D. Enable push protection

Answer: D

Explanation:

The best proactive control is push protection. It scans for secrets during a git push and blocks the commit before it enters the repository.

Other options (like CODEOWNERS or security managers) help with oversight but do not prevent secret leaks.

Making a repo public would increase the risk, not reduce it.

Reference: GitHub Docs – Enabling Push Protection

Question: 58

– [Configure GitHub Actions Workflows]

As a repository owner, you do not want to run a GitHub Actions workflow when changes are made to any .txt or markdown files. How would you adjust the event trigger for a pull request that targets the main branch? (Each answer presents part of the solution. Choose three.)

on:

pull_request:

branches: [main]

A. - '/*.md'

B. - '/*.txt'

C. paths:

D. paths-ignore:

E. - 'docs/*.md'

Answer: A, B, D

Explanation:

To exclude .txt and .md files from triggering workflows on pull requests to the main branch:

on: defines the event (e.g., pull_request)

pull_request: is the trigger

paths-ignore: is the key used to ignore file patterns

Example YAML:

yaml

CopyEdit

```
on:
  pull_request:
    branches:
      - main
    paths-ignore:
      - '*.md'
      - '*.txt'
```

Using paths: would include only specific files instead — not exclude. paths-ignore: is correct here.

Reference: GitHub Docs – Workflow Syntax for GitHub Actions

Question: 59

– [Use Code Scanning with CodeQL]

What does code scanning do?

A. It contacts maintainers to ask them to create security advisories if a vulnerability is found

B. It prevents code pushes with vulnerabilities as a pre-receive hook

C. It analyzes a GitHub repository to find security vulnerabilities

D. It scans your entire Git history on branches present in your GitHub repository for any secrets

Answer: C

Explanation:

Code scanning is a static analysis feature that examines your source code to identify security vulnerabilities and coding errors. It runs either on every push, pull request, or a scheduled time depending on the workflow configuration. It does not automatically contact maintainers, scan full Git history, or block pushes unless explicitly configured to do so.

Reference: GitHub Docs – About Code Scanning

Question: 60

– [Configure GitHub Advanced Security Tools in GitHub Enterprise]

As a developer, you need to configure a code scanning workflow for a repository where GitHub Advanced Security is enabled. What minimum repository permission do you need?

A. Write

B. None

C. Admin

D. Read

Answer: A

Explanation:

To create or modify a code scanning workflow file (typically under .github/workflows/codeql- analysis.yml), you must have Write access to the repository.

Write permission allows you to commit the workflow file, which is required to run or configure code scanning using GitHub Actions.

Reference: GitHub Docs – Repository Permissions and Workflow Setup

Question: 61

– [Describe GHAS Security Features and Functionality]

Which alerts do you see in the repository's Security tab? (Each answer presents part of the solution. Choose three.)

A. Repository permissions

B. Secret scanning alerts

C. Dependabot alerts

D. Security status alerts

E. Code scanning alerts

Answer: B, C, E

Explanation:

In a repository's Security tab, you can view:

Secret scanning alerts: Exposed credentials or tokens

Dependabot alerts: Vulnerable dependencies from the advisory database

Code scanning alerts: Vulnerabilities in code detected via static analysis (e.g., CodeQL)

You won't see general "security status alerts" (not a formal category) or permission-related alerts here.

Reference: GitHub Docs – Understanding the Security Tab

Question: 62

– [Use Code Scanning with CodeQL]

Which CodeQL query suite provides queries of lower severity than the default query suite?

A. github/codeql-go/ql/src@main

B. github/codeql/cpp/ql/src@main

C. security-extended

Answer: C

Explanation:

The security-extended query suite includes additional CodeQL queries that detect lower severity issues than those in the default security-and-quality suite.

It's often used when projects want broader visibility into code hygiene and potential weak spots beyond critical vulnerabilities.

The other options listed are paths to language packs, not query suites themselves.

Reference: GitHub Docs – CodeQL Query Suite Types

Question: 63

– [Configure and Use Dependency Management]

Which of the following options would close a Dependabot alert?

A. Creating a pull request to resolve the vulnerability that will be approved and merged

B. Viewing the Dependabot alert on the Dependabot alerts tab of your repository

C. Viewing the dependency graph

D. Leaving the repository in its current state

Answer: A

Explanation:

A Dependabot alert is only marked as resolved when the related vulnerability is no longer present in your code — specifically after you merge a pull request that updates the vulnerable dependency.

Simply viewing alerts or graphs does not affect their status. Ignoring the alert by leaving the repo unchanged keeps the vulnerability active and unresolved.

Reference: GitHub Docs – Managing Dependabot Security Updates

Question: 64

– [Configure and Use Dependency Management]

What are Dependabot security updates?

A. Automated pull requests that help you update dependencies that have known vulnerabilities

B. Automated pull requests that keep your dependencies updated, even when they don't have any vulnerabilities

C. Automated pull requests to update the manifest to the latest version of the dependency

D. Compatibility scores to let you know whether updating a dependency could cause breaking changes to your

project

Answer: A

Explanation:

Dependabot security updates are automated pull requests triggered when GitHub detects a vulnerability in a dependency listed in your manifest or lockfile. These PRs upgrade the dependency to the minimum safe version that fixes the vulnerability.
This is separate from regular updates (which keep versions current even if not vulnerable).

Reference: GitHub Docs – About Dependabot Security Updates

Question: 65

– [Use Code Scanning with CodeQL]
Which of the following steps should you follow to integrate CodeQL into a third-party continuous integration system? (Each answer presents part of the solution. Choose three.)

A. Process alerts
B. Analyze code
C. Upload scan results
D. Install the CLI
E. Write queries

Answer: B, C, D

Explanation:

When integrating CodeQL outside of GitHub Actions (e.g., in Jenkins, CircleCI):

Install the CLI: Needed to run CodeQL commands.

Analyze code: Perform the CodeQL analysis on your project with the CLI.

Upload scan results: Export the results in SARIF format and use GitHub's API to upload them to your repo's security tab.

You don't need to write custom queries unless extending functionality. "Processing alerts" happens after GitHub receives the results.

Reference: GitHub Docs – Using CodeQL with 3rd Party CI Systems

Question: 66

– [Use Code Scanning with CodeQL]

Which syntax in a query suite tells CodeQL to look for one or more specified .ql files?

A. query

B. qlpack

C. qls

Answer: A

Explanation:

In a query suite (a .qls file), the **query** key is used to specify the paths to one or more .ql files that should be included in the suite.

Example:

D. query: path/to/query.ql

qls is the file format.

qlpack is used for packaging queries, not in suite syntax.

Reference: GitHub Docs – CodeQL Query Suite Syntax

Question: 67

– [Configure and Use Dependency Management]

Which security feature shows a vulnerable dependency in a pull request?

A. Dependency graph

B. Dependency review

C. Dependabot alert

D. The repository's Security tab

Answer: B

Explanation:

Dependency review runs as part of a pull request and shows which dependencies are being added, removed, or changed — and highlights vulnerabilities associated with any added packages.

It works in real-time and is specifically designed for use during pull request workflows.

The dependency graph is an overview, Dependabot alerts notify post-merge, and the Security tab shows the aggregated alert list.

Reference: GitHub Docs – About Dependency Review

Question: 68

E. [Configure and Use Secret Scanning]

A secret scanning alert should be closed as "used in tests" when a secret is:

A. In the readme.md file.

B. In a test file.

C. Solely used for tests.

D. Not a secret in the production environment.

Answer: C

Explanation:

If a secret is intentionally used in a test environment and poses no real-world security risk, you may close the alert with the reason "used in tests". This helps reduce noise and clarify that the alert was reviewed and accepted as non-critical.

Just being in a test file isn't enough unless its purpose is purely for testing.

Reference: GitHub Docs – Managing Secret Scanning Alerts

Question: 69

– [Use Code Scanning with CodeQL]

The autobuild step in the CodeQL workflow has failed. What should you do?

A. Remove specific build steps.

B. Compile the source code.

C. Remove the autobuild step from your code scanning workflow and add specific build steps.

D. Use CodeQL, which implicitly detects the supported languages in your code base.

Answer: C

Explanation:

If autobuild fails (which attempts to automatically detect how to build your project), you should disable it in your

workflow and replace it with explicit build commands, using steps like run: make or run: ./gradlew build.

This ensures CodeQL can still extract and analyze the code correctly.

Reference: GitHub Docs – CodeQL Build Configurations for Compiled Languages

Question: 70

– [Configure and Use Dependency Management]

In the pull request, how can developers avoid adding new dependencies with known vulnerabilities?

A. Enable Dependabot alerts.

B. Add Dependabot rules.

C. Add a workflow with the dependency review action.

D. Enable Dependabot security updates.

Answer: C

Explanation:

To detect and block vulnerable dependencies before merge, developers should use the Dependency Review GitHub Action in their pull request workflows. It scans all proposed dependency changes and flags any packages with known vulnerabilities.

This is a preventative measure during development, unlike Dependabot, which reacts after the fact.

Reference: GitHub Docs – Dependency Review Action

Question: 71

– [Configure and Use Secret Scanning]

Where in the repository can you give additional users access to secret scanning alerts?

A. Security

B. Settings

C. Secrets

D. Insights

Answer: B

Explanation:

To grant specific users access to view and manage secret scanning alerts, you do this via the Settings tab of the repository. From there, under the "Code security and analysis" section, you can add individuals or teams with roles such as security manager.

The Security tab only displays alerts; access control is handled in Settings.

Reference: GitHub Docs – Granting Access to Secret Scanning Alerts

Question: 72

– [Configure and Use Dependency Management]

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

Answer: A

Explanation:

By default, users with Write, Maintain, or Admin permissions will receive notifications for new Dependabot alerts. However, Write permission is the minimum level needed to be automatically notified. Users with only Read access do not receive alerts unless added explicitly.

Reference: GitHub Docs – Dependabot Alerts Notification Scope

Question: 73

– [Configure and Use Secret Scanning]

Which patterns are secret scanning validity checks available to?

A. High entropy strings

B. Custom patterns

C. Partner patterns

D. Push protection patterns

Answer: C

Explanation:

Validity checks — where GitHub verifies if a secret is still active — are available for partner patterns only. These are secrets issued by GitHub's trusted partners (like AWS, Slack, etc.) and have APIs for GitHub to validate token activity status.

Custom patterns and high entropy patterns do not support automated validity checks.

Reference: GitHub Docs – Secret Validation for Partner Patterns

Question: 74

– [Configure and Use Dependency Management]
A repository's dependency graph includes:

A. Dependencies parsed from a repository's manifest and lock files.

B. Annotated code scanning alerts from your repository's dependencies.

C. A summary of the dependencies used in your organization's repositories.

D. Dependencies from all your repositories.

Answer: A

Explanation:

The dependency graph in a repository is built by parsing manifest and lock files (like package.json, pom.xml, requirements.txt). It helps GitHub detect dependencies and cross-reference them with known vulnerability databases for alerting.

It is specific to each repository and does not show org-wide or cross-repo summaries.

Reference: GitHub Docs – Understanding the Dependency Graph

Question: 75

– [Configure and Use Secret Scanning]

What filter or sort settings can be used to prioritize the secret scanning alerts that present the most risk?

A. Sort to display the oldest first

B. Sort to display the newest first

C. Filter to display active secrets

D. Select only the custom patterns

Answer: C

Explanation:

The best way to prioritize secret scanning alerts is to filter by active secrets — these are secrets GitHub has confirmed are still valid and could be exploited. This allows security teams to focus on high-risk exposures that require immediate attention.

Sorting by time or filtering by custom patterns won't help with risk prioritization directly.

Reference: GitHub Docs – Filtering Secret Scanning Alerts