MICROSOFT GH-500 Exam Questions

Total Questions: 70+
Demo Questions: 15
Version: Updated for 2025

Prepared and Verified by Cert Empire – Your Trusted IT Certification Partner

For Access to the full set of Updated Questions – Visit:
MICROSOFT GH-500 Exam Questions by Cert Empire

Question: 1

- 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:

Custom patterns are an advanced configuration of the secret scanning feature. The base secret scanning functionality must be active before you can extend it with custom rules. Enabling secret scanning is the foundational step that allows the system to scan for secrets. Once enabled, you can then augment its capabilities by defining custom patterns for the repository, organization, or enterprise. Without the core feature being enabled, there is no scanning engine to which custom patterns can be applied.

Why Incorrect Options are Wrong:

A. Change the repository visibility to Internal: Secret scanning and custom patterns are available for various repository visibilities (private, internal, and public on Enterprise), not just 'Internal'. This is not a required prerequisite step.
B. Close other secret scanning alerts: The state of existing alerts is irrelevant to the configuration of new scanning patterns. Alert management and pattern definition are separate, independent administrative tasks.
C. Specify additional match criteria: This is an optional step during the process of defining a custom pattern to reduce false positives, not a prerequisite before you can start defining one.

References:

1. GitHub Docs, "Defining custom patterns for secret scanning." Under the "Prerequisites" section, the documentation explicitly states: "Before you can define a custom pattern for your repository, you must enable secret scanning for your repository." This document also confirms that custom patterns are a feature of GitHub Advanced Security.
2. GitHub Docs, "Configuring secret scanning for your repositories." This page details the process of enabling the feature, which is the necessary first step before any advanced configuration, such as adding custom patterns, can be performed. It establishes enabling the feature as the primary action.

3. GitHub Docs, "About secret scanning." This document provides an overview of the feature, explaining that it is enabled at the repository or organization level. This foundational enablement is a prerequisite for all subsequent configurations, including custom patterns.

Question: 2

- 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:

CodeQL analysis for code scanning can be executed in two primary environments. The most common method is natively within GitHub through a GitHub Actions workflow, which automates the process of checking out code, running the analysis, and uploading the results. For organizations that use other CI/CD platforms, GitHub provides the CodeQL runner, a command-line tool that allows the integration of CodeQL analysis into external continuous integration (CI) systems like Jenkins, Azure Pipelines, or CircleCI. These two methods provide flexibility for integrating security scanning directly into the development pipeline, regardless of the specific CI/CD tooling used.

Why Incorrect Options are Wrong:

A. In a third-party Git repository: CodeQL analysis is not run within a third-party repository system (like GitLab or Bitbucket) itself. The code must be checked out and analyzed in a separate compute environment, such as a CI system.
D. In the Files changed tab of the pull request: This tab is used to display the results and alerts from a completed CodeQL scan, not to initiate or run the analysis.

References:

1. GitHub Docs, "About code scanning with CodeQL." This document outlines the primary methods for running CodeQL. It states, "You run CodeQL code scanning on a repository by adding a GitHub Actions workflow... Alternatively, if you have a complex build, you can run the CodeQL runner directly in your third-party CI system." This directly supports options B and C.
2. GitHub Docs, "Configuring code scanning for a repository." This guide details the setup process, emphasizing the use of GitHub Actions. Section "Running code scanning using GitHub Actions" states, "You can run code scanning on GitHub using GitHub Actions. You enable code scanning by adding a workflow to your repository." This confirms option B.
3. GitHub Docs, "Running CodeQL code scanning in your CI system." This document is dedicated

to using CodeQL outside of GitHub Actions. It begins, "If you don't use GitHub Actions, you can run CodeQL analysis in a third-party continuous integration (CI) system using the CodeQL runner." This confirms option C.

Question: 3

- 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:

Code scanning results generated by CodeQL analysis are displayed as alerts within the GitHub repository. These alerts are aggregated under the "Security" tab, specifically in the "Code scanning alerts" section. This interface allows users to view, triage, and manage potential vulnerabilities and errors detected in the code. Each alert provides context, including the location of the issue in the code, a description of the vulnerability, and often, guidance on how to fix it. This centralized view is the primary location for interacting with code scanning findings.

Why Incorrect Options are Wrong:

B. A CodeQL database is an intermediate artifact containing a queryable representation of the code; it is used for analysis, not for viewing final results.
C. A CodeQL query pack is a collection of queries used as an input to the analysis process to find vulnerabilities, not the output itself.
D. Security advisories are used to privately discuss, fix, and publish information about vulnerabilities, a distinct process from viewing automated scan results.

References:

1. GitHub Docs, "About code scanning alerts." This document states, "You can view alerts from code scanning in the Security tab of your repository... Each alert shows you the problem, the name of the tool that identified it, and the lines of code that triggered the alert."
2. GitHub Docs, "Managing code scanning alerts for your repository." This guide details the process: "You can find and fix alerts for potential vulnerabilities and errors in your repository's code. 1. On GitHub.com, navigate to the main page of the repository. 2. Under your repository name, click Security. 3. In the left sidebar, click Code scanning alerts."
3. GitHub Docs, "About code scanning with CodeQL." This document explains the components of the process, clarifying that "CodeQL analyzes the CodeQL database and the results are displayed as code scanning alerts in GitHub." This distinguishes the database (B) from the alerts

(A).

Question: 4

- 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:

In CodeQL, a query suite is a collection of queries designed to be run together. The definition for a query suite is stored in a file with the .qls extension. These files use a specific YAML-based format to specify which queries to include or exclude from the analysis. This allows for the creation of custom, targeted sets of security checks. For example, a .qls file can reference individual queries, directories containing queries, or other query suite files.

Why Incorrect Options are Wrong:

A. .yml: This extension is used for GitHub Actions workflow files, which define the steps to run CodeQL analysis, but not the query suite definition itself.
B. .ql: This is the file extension for an individual CodeQL query, which contains the logic to identify a specific pattern or vulnerability.
C. .qll: This extension denotes a CodeQL library file. These files contain reusable modules, classes, and predicates that are imported by .ql files.

References:

1. GitHub Docs, "Creating CodeQL query suites." This document explicitly states, "Query suite definitions are stored in files with the extension .qls." It further details the YAML-based structure used within these files to define the collection of queries.
Source: GitHub, CodeQL Code Scanning Documentation. Section: "Creating CodeQL query suites."
2. GitHub Docs, "About CodeQL queries." This page describes the different files used in CodeQL analysis, clarifying the roles of .ql and .qll files. It defines a .ql file as a query and a .qll file as a "query library" used to help write queries.
Source: GitHub, CodeQL Code Scanning Documentation. Section: "About CodeQL queries."

Question: 5

- 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:

Modifying the code scanning severity threshold for pull request checks is a repository-level security setting. This action is performed under the "Code security and analysis" section of a repository's settings. According to the official GitHub permission model, only users with the 'admin' role have the authority to access and change these critical security configurations. This level of control is reserved for administrators to ensure that the repository's security policies are managed securely and consistently.

Why Incorrect Options are Wrong:

A. Maintain: The 'maintain' role can manage some repository settings, such as topics and collaborators, but lacks the permissions to alter core security and analysis configurations.
B. Write: The 'write' role grants permissions to contribute code, create branches, and manage pull requests, but it does not include access to repository settings.
C. Triage: The 'triage' role is the most limited, allowing users to manage issues and pull requests without any write access or ability to change settings.

References:

1. GitHub Docs, "Repository roles for an organization": This document explicitly states that the admin role is required to "Manage repository security and analysis settings." The maintain, write, and triage roles do not have this permission. (See the permissions table under the "Permissions for each role" section).
2. GitHub Docs, "Configuring code scanning": This guide details the process for setting up and customizing code scanning. It notes that to access the configuration page (Settings Code security and analysis), a user must have admin permissions for the repository. (See the "Prerequisites" or introductory sections of the configuration guides).

Question: 6

- 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:

A, D

Explanation:

When using an advanced setup for CodeQL, you can specify additional queries to run alongside the default suite by creating a configuration file. This file uses the queries key to define a list of specific .ql files or directories to run. Alternatively, the packs key can be used to include one or more CodeQL query packs, which are collections of queries, often used for custom or specialized security checks. Both methods allow for the extension of the default code scanning analysis.

Why Incorrect Options are Wrong:

B. github/codeql is the repository containing the standard CodeQL queries, not a configuration key for adding custom ones.
C. Scope is not a recognized key in the CodeQL configuration file for specifying additional queries.

References:

1. GitHub Docs. (n.d.). Customizing your advanced setup for code scanning. Retrieved from https ://docs.github.com/en/code-security/code-scanning/creating-an-advanced-setup-for-code-scannin g/customizing-your-advanced-setup-for-code-scanning
For option D (Queries): In the section "Specifying queries to run," the documentation states, "You can use queries to specify which queries to run in your configuration file." It provides examples of using the uses keyword to point to a .ql file, a directory containing queries, or a query suite definition file.
For option A (Packs): In the section "Running additional query packs," the documentation explains, "To add one or more CodeQL query packs, add a packs entry to your configuration file." It shows how to list packs to be used in the analysis.

Question: 7

- 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:

To integrate a SARIF-compatible static analysis tool with GitHub code scanning using GitHub Actions, the workflow must be explicitly configured with multiple steps. After a step runs the analysis tool and generates a SARIF file, a distinct, final step is required to upload this file to GitHub. This is accomplished using the github/codeql-action/upload-sarif action. This action takes the path to the SARIF file as input and uploads its findings to the repository's security tab, populating the code scanning alerts. This process is required for all third-party static analysis tools.

Why Incorrect Options are Wrong:

B. The GitHub Actions runner only executes the jobs defined in a workflow; it does not have default behaviors like automatically uploading specific file types.
C. This is only true for the native CodeQL action's analyze step. The question refers to any "SARIF-compatible tool," which includes third-party tools that require a separate upload step.
D. While the GitHub CLI (gh) can upload SARIF files, the idiomatic and recommended method within a GitHub Actions workflow is to use the dedicated upload-sarif action.

References:

1. GitHub Docs, "Uploading a SARIF file to GitHub." This document explicitly states, "You can use the github/codeql-action/upload-sarif action to upload a SARIF file... This is useful when you use a static analysis tool other than CodeQL." It provides a clear YAML example with a final step dedicated to the upload:
yaml
- name: Upload analysis results to GitHub
uses: github/codeql-action/upload-sarif@v3
with:
sariffile: results.sarif

Source: GitHub Enterprise Cloud Documentation, "Code security," "Code scanning," "Integrating with code scanning," "Uploading a SARIF file to GitHub."
2. GitHub Docs, "About SARIF file uploads for code scanning." This page clarifies the general requirement: "You need to run your SARIF-compatible analysis tool in a GitHub Actions workflow... and include a step that uploads the results.sarif file."
Source: GitHub Enterprise Cloud Documentation, "Code security," "Code scanning," "Integrating with code scanning," "About SARIF file uploads for code scanning."

Question: 8

- 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 a code scanning alert identifies a data flow problem, it means potentially untrusted data is being used in an unsafe way. To understand the vulnerability, you need to trace the path of this data from its origin (the source) to where it's used unsafely (the sink). The GitHub interface provides a specific control for this purpose. By clicking Show paths, you can expand the alert details to see a step-by-step visualization of the data's journey through the codebase, providing the necessary context to remediate the issue.

Why Incorrect Options are Wrong:

B. Security: This is the top-level tab in a repository's navigation where you access all security-related features, not a specific action taken within an alert.
C. Code scanning alerts: This is the link under the "Security" tab that takes you to the list of all alerts, not the control used to get more context on a single, specific alert.

References:

1. GitHub Docs, "Managing code scanning alerts for your repository." Under the section "Viewing the details of an alert," the documentation states: "For alerts that highlight a data-flow problem, you can also view the path from the data source to the sink. To view the data flow path, click Show paths." This directly confirms the function of the "Show paths" link for data flow analysis.
2. GitHub Docs, "About code scanning alerts." In the section "About data flow analysis," it explains: "Data flow analysis finds potential security issues in code by tracking the flow of data from a source... to a sink... Code scanning shows you how the data travels from the source to the sink in the alert details." This establishes the concept that viewing the path is the primary way to get context.

Question: 9

- 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:

GitHub secret scanning is a native security feature that operates specifically on repositories hosted on the GitHub platform (GitHub.com and GitHub Enterprise). It automatically scans the entire Git history of all branches within a repository to find accidentally committed secrets, such as tokens and private keys. The scope of the scan is confined to the content of the GitHub repository itself, making it a repository-centric security tool.

Why Incorrect Options are Wrong:

A. Secret scanning is a GitHub feature that scans repository content, not the continuous integration system's infrastructure or configuration itself.
B. The feature is specific to repositories hosted on GitHub. It does not scan any arbitrary Git repository hosted on other platforms like GitLab or a private server.
D. Secret scanning finds credentials for external services that are located within a repository; it does not scan the external services themselves.

References:

1. GitHub Docs, "About secret scanning." This document explicitly states, "Secret scanning scans your entire Git history on all branches present in your GitHub repository for secrets." This confirms the feature's scope is the GitHub repository.
Source: https://docs.github.com/en/code-security/secret-scanning/about-secret-scanning, Section: "About secret scanning."
2. Microsoft Learn, "Configure secret scanning," GH-500 Learning Path. The official training material for the exam describes the feature's function: "GitHub scans repositories for known types of secrets to prevent fraudulent use of secrets that were committed by accident." The context is always a repository on GitHub.
Source: https://learn.microsoft.com/en-us/training/modules/configure-secret-scanning-in-your-repo/2-what-is-secret-scanning, Paragraph 1.
3. GitHub Docs, "Managing security and analysis settings for your repository." The configuration

process for secret scanning is performed within the settings of a specific GitHub repository, reinforcing that its operational boundary is the repository.
Source: https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/enabling-features-for-your-repository/managing-security-and-analysis-settings-for-your-repository, Section: "Allowing or disallowing features for your private repository."

Question: 10

- 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 are powered by the GitHub Advisory Database. When a vulnerability is identified, the resulting alert uses standardized formats to describe it. The Common Vulnerabilities and Exposures (CVE) system provides a unique identifier for each specific, publicly known cybersecurity vulnerability. Additionally, the Common Weakness Enumeration (CWE) is used to classify the type of software or hardware weakness that led to the vulnerability (e.g., CWE-79 for Cross-site Scripting). Both formats are integral to the information presented in a Dependabot alert, providing both a unique ID and a categorical description.

Why Incorrect Options are Wrong:

B. Exploit Prediction Scoring System (EPSS): This is a scoring system that estimates the probability of a vulnerability being exploited. It provides a metric for prioritization, not a format for describing the vulnerability itself.
D. Vulnerability Exploitability eXchange (VEX): VEX is a format used to communicate the status of a vulnerability within a specific product (e.g., "not affected"), rather than describing the initial vulnerability alert.

References:

1. GitHub Docs, "About the GitHub Advisory Database": This document explicitly states, "Each security advisory includes information about the vulnerability, which may include... a CVE (Common Vulnerabilities and Exposures) identifier... and one or more Common Weakness Enumeration (CWE) identifiers." This directly confirms the use of both CVE and CWE.
Source: GitHub, Inc. (2024). About the GitHub Advisory Database. GitHub Docs. Retrieved from https://docs.github.com/en/code-security/security-advisories/global-security-advisories/about-the-github-advisory-database (See the section "About security advisories").
2. GitHub Docs, "About Dependabot alerts": This page clarifies that Dependabot alerts are based on vulnerabilities from the GitHub Advisory Database, which uses CVE identifiers.

Source: GitHub, Inc. (2024). About Dependabot alerts. GitHub Docs. Retrieved from https://docs.github.com/en/code-security/dependabot/dependabot-alerts/about-dependabot-alerts (See the section "Detection of insecure dependencies").

Question: 11

- 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:

To request a Common Vulnerabilities and Exposures (CVE) identification number, you must first create a repository security advisory. According to official GitHub documentation, creating and managing security advisories is a privileged action that requires administrator permissions for the repository. This ensures that only authorized individuals can publicly disclose and manage vulnerabilities associated with the repository's codebase. The process of requesting a CVE is an integral part of managing the advisory, and thus inherits the same permission requirement.

Why Incorrect Options are Wrong:

A. Maintain: Users with Maintain permissions can manage many repository settings but lack the authority to create security advisories or request CVEs.
C. Triage: The Triage role is limited to managing issues and pull requests and does not include permissions for security management features.
D. Write: The Write role allows users to contribute code and manage pull requests but does not grant access to repository security settings like advisories.

References:

1. GitHub Docs, "Creating a repository security advisory." Under the "Prerequisites" section, it explicitly states: "You must have admin permissions for the repository." This prerequisite applies to the entire advisory creation process, which includes requesting a CVE.
Source: GitHub Official Documentation. Retrieved from https://docs.github.com/en/code-security/security-advisories/repository-security-advisories/creating-a-repository-security-advisory#prerequisites
2. GitHub Docs, "About GitHub Security Advisories for repositories." This document clarifies the roles involved, stating, "Repository owners and security managers have admin permissions for security advisories in a repository." This reinforces that administrative-level access is the

foundation for managing advisories.
Source: GitHub Official Documentation. Retrieved from https://docs.github.com/en/code-security/security-advisories/repository-security-advisories/about-github-security-advisories-for-repositories
3. GitHub Docs, "Permission levels for a personal account repository." The repository roles and permissions table confirms that actions related to "Manage security advisories" are exclusively available to the Admin role.
Source: GitHub Official Documentation. Retrieved from https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-personal-account-on-github/managing-access-to-your-personal-repositories/permission-levels-for-a-personal-account-repository

Question: 12

- 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 underlying issue in the code is resolved. This process involves a developer committing a fix to the branch where the alert was detected. When the branch is re-scanned (typically after the commit is pushed, often within a pull request), the CodeQL analysis will no longer detect the vulnerability. Upon confirming the absence of the issue in the latest analysis for that branch, GitHub marks the alert as fixed and closes it.

Why Incorrect Options are Wrong:

A. Triaging an alert involves manually reviewing and dismissing it (e.g., as a false positive or acceptable risk), which is a different action from fixing the code to resolve it.
B. Data-flow analysis is a technique used by CodeQL to find potential security issues; it is part of the detection process, not the resolution or closure process.
C. Clicking an alert within a pull request is a navigational action for viewing its details; it does not alter the code or the status of the alert.

References:

1. GitHub Docs, "Managing code scanning alerts for your repository." Under the section "Fixing an alert," the documentation states, "GitHub automatically closes an alert if you fix the code in a pull request... GitHub closes an alert when code scanning determines that the code with the potential vulnerability is no longer present in the latest analysis for a branch."
2. Microsoft Learn, "Manage code scanning alerts in GitHub." In the "Fix code scanning alerts" unit, it is specified that "GitHub closes an alert automatically when you fix the code that triggered it. To fix an alert, you need to commit your changes to the branch where the alert was found."
3. GitHub Docs, "About code scanning alerts." In the section "About the status and details of alerts," it clarifies the lifecycle: "An alert is usually closed when a user fixes the code that triggered the alert and pushes their changes to the branch that is being scanned."

Question: 13

- 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:

A, B

Explanation:

When defining a custom pattern for GitHub secret scanning, two fields are mandatory for its creation. You must provide a unique Name to identify and manage the pattern. You must also define the Secret format using a regular expression (regex). This regex is the core of the pattern, specifying the structure of the secret that the scanning engine will search for. These two components are the minimum required details to save a new custom pattern at the organization or enterprise level.

Why Incorrect Options are Wrong:

C. A list of repositories to scan
Specifying repositories is part of enabling or configuring a scan, not defining the pattern itself.
D. Additional match requirements for the secret format
These are optional settings, such as "surrounding content," used to refine the main pattern and reduce false positives, not a mandatory creation detail.

References:

1. GitHub Docs, "Defining custom patterns for secret scanning." This official documentation outlines the steps for creating a custom pattern. In the section "Creating a custom pattern," it explicitly lists "Name" and "Secret format (regex)" as required fields. It also describes "Additional match requirements" as an optional step to refine results. The process described does not include selecting repositories during pattern creation.
Reference Location: Navigate to the "Creating a custom pattern" section. The input fields in the UI steps clearly distinguish between required and optional information.
2. GitHub Docs, "About secret scanning." This document provides an overview of the feature and distinguishes between defining patterns and enabling them for repositories. It clarifies that patterns are defined at the organization/enterprise level before being applied to specific repositories or all repositories.

Reference Location: See the section on "Defining custom patterns for secret scanning."

Question: 14

- 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 receive alerts for detected vulnerabilities, you must enable the "Dependabot alerts" feature. Within your personal account's "Code security and analysis" settings, selecting "Enable all" for Dependabot alerts activates this feature for all repositories you own. This action configures GitHub to automatically scan your dependencies, identify vulnerabilities, and generate alerts. While the Dependency graph is a prerequisite, it only identifies dependencies; it does not generate the vulnerability alerts itself.

Why Incorrect Options are Wrong:

A. Enable all in existing repositories: This option is too generic. It does not specify which security feature (e.g., Dependabot alerts, Secret scanning, Code scanning) should be enabled.
B. Enable by default for new public repositories: This setting only applies to newly created public repositories and would not activate alerts for any existing or private repositories.
D. Enable all for Dependency graph: Enabling the Dependency graph is a necessary prerequisite, as it identifies the project's dependencies. However, it does not generate vulnerability alerts on its own; that is the specific function of Dependabot alerts.

References:

1. GitHub Docs, "Managing security and analysis settings for your user account."
Location: Section "Enabling or disabling features for existing repositories."
Content: This document explicitly outlines the steps to manage security features. It states, "Under 'Code security and analysis', find the feature you want to manage... To the right of the feature, click Disable all or Enable all." It lists "Dependabot alerts" as a distinct feature that can be enabled, which is the direct action required to receive vulnerability alerts.
2. GitHub Docs, "About Dependabot alerts."

Location: Introduction section.
Content: This page clarifies the relationship between the dependency graph and alerts: "Dependabot alerts are generated when GitHub detects that a repository uses a dependency with a known vulnerability... The dependency graph is a prerequisite for Dependabot alerts." This confirms that enabling the graph (Option D) is insufficient and that enabling "Dependabot alerts" (Option C) is the correct, specific action.

Question: 15

- 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:

GitHub Advanced Security (GHAS) is a suite of security features. The question specifically asks about three of its core components:

1. Code scanning analyzes your code to find security vulnerabilities and coding errors.
2. Secret scanning detects secrets, such as API keys and tokens, that have been checked into the repository.
3. Dependency review allows you to visualize dependency changes and their security impact directly within a pull request, helping you understand the full effect before merging.

Option A accurately summarizes the primary function of each of these three distinct features.

Why Incorrect Options are Wrong:

B. This describes private vulnerability reporting, a feature that allows security researchers to disclose vulnerabilities to repository maintainers confidentially, which is separate from the listed GHAS tools.

C. This describes Dependabot alerts, which notify you about existing dependencies with known vulnerabilities in your repository, but it doesn't cover code scanning or secret scanning.

D. This describes Dependabot security and version updates, which automatically create pull requests to update dependencies, a different function from the three features mentioned in the question.

References:

1. GitHub Docs, "About GitHub Advanced Security." This document provides an overview, stating, "GitHub Advanced Security features are available for enterprise accounts on GitHub Enterprise Cloud and GitHub Enterprise Server... It provides additional features... such as code scanning, secret scanning, and dependency review."
Source: https://docs.github.com/en/get-started/learning-about-github/about-github-advanced-security
2. GitHub Docs, "About code scanning." Section: "About code scanning." It states, "Code scanning is a feature that you use to analyze the code in a GitHub repository to find security vulnerabilities and coding errors."
Source: https://docs.github.com/en/code-security/code-scanning/introduction-to-code-scanning/about-code-scanning
3. GitHub Docs, "About secret scanning." Section: "About secret scanning." It explains, "Secret scanning helps protect you by searching repositories for known types of secrets... GitHub scans the content for secrets."
Source: https://docs.github.com/en/code-security/secret-scanning/about-secret-scanning
4. GitHub Docs, "About dependency review." Section: "What is dependency review?" It details, "Dependency review helps you understand dependency changes and the security impact of these changes at every pull request... You can clearly see what's changed in a pull request."
Source: https://docs.github.com/en/code-security/supply-chain-security/understanding-your-software-supply-chain/about-dependency-review