GH-500 Practice Exam Questions, GH-500 Examcollection Free Dumps

BONUS!!! Download part of CramPDF GH-500 dumps for free: https://drive.google.com/open?id=1CkTvrdSrElTOrTjxqj4s-mpZpudYL-YF

We are committed to designing a kind of scientific GH-500 study material to balance your business and study schedule. With our GH-500 exam guide, all your learning process includes 20-30 hours. As long as you spare one or two hours a day to study with our laTest GH-500 Quiz prep, we assure that you will have a good command of the relevant knowledge before taking the GH-500 exam. What you need to do is to follow the GH-500 exam guide system at the pace you prefer as well as keep learning step by step.

By clearing different Microsoft exams, you can easily land your dream job. If you are looking to find high paying jobs, then Microsoft certifications can help you get the job in the highly reputable organization. Our GH-500 exam materials give real exam environment with multiple learning tools that allow you to do a selective study and will help you to get the job that you are looking for. Moreover, we also provide 100% money back guarantee on our GH-500 Exam Materials, and you will be able to pass the GH-500 exam in short time without facing any troubles.

>> GH-500 Practice Exam Questions <<

Microsoft GH-500 Practice Exam Questions offer you accurate Examcollection Free Dumps to pass GitHub Advanced Security exam

For the complete GitHub Advanced Security exam preparation and success, the CramPDF GH-500 exam practice test questions are the best choice. With the Microsoft GH-500 Exam Questions, you will get everything that you need to learn, prepare and succeed in the GitHub Advanced Security certification exam. You must add Microsoft GH-500 Exam Questions in your preparation and should not ignore them.

Microsoft GH-500 Exam Syllabus Topics:

Topic	Details
Topic 1	• Describe GitHub Advanced Security best practices, results, and how to take corrective measures: This section evaluates skills of Security Managers and Development Team Leads in effectively handling GHAS results and applying best practices. It includes using Common Vulnerabilities and Exposures (CVE) and Common Weakness Enumeration (CWE) identifiers to describe alerts and suggest remediation, decision-making processes for closing or dismissing alerts including documentation and data-based decisions, understanding default CodeQL query suites, how CodeQL analyzes compiled versus interpreted languages, the roles and responsibilities of development and security teams in workflows, adjusting severity thresholds for code scanning pull request status checks, prioritizing secret scanning remediation with filters, enforcing CodeQL and Dependency Review workflows via repository rulesets, and configuring code scanning, secret scanning, and dependency analysis to detect and remediate vulnerabilities earlier in the development lifecycle, such as during pull requests or by enabling push protection.

Topic 2	• Configure and use Dependabot and Dependency Review: Focused on Software Engineers and Vulnerability Management Specialists, this section describes tools for managing vulnerabilities in dependencies. Candidates learn about the dependency graph and how it is generated, the concept and format of the Software Bill of Materials (SBOM), definitions of dependency vulnerabilities, Dependabot alerts and security updates, and Dependency Review functionality. It covers how alerts are generated based on the dependency graph and GitHub Advisory Database, differences between Dependabot and Dependency Review, enabling and configuring these tools in private repositories and organizations, default alert settings, required permissions, creating Dependabot configuration files and rules to auto-dismiss alerts, setting up Dependency Review workflows including license checks and severity thresholds, configuring notifications, identifying vulnerabilities from alerts and pull requests, enabling security updates, and taking remediation actions including testing and merging pull requests.
Topic 3	• Describe the GHAS security features and functionality: This section of the exam measures skills of Security Engineers and Software Developers and covers understanding the role of GitHub Advanced Security (GHAS) features within the overall security ecosystem. Candidates learn to differentiate security features available automatically for open source projects versus those unlocked when GHAS is paired with GitHub Enterprise Cloud (GHEC) or GitHub Enterprise Server (GHES). The domain includes knowledge of Security Overview dashboards, the distinctions between secret scanning and code scanning, and how secret scanning, code scanning, and Dependabot work together to secure the software development lifecycle. It also covers scenarios contrasting isolated security reviews with integrated security throughout the development lifecycle, how vulnerable dependencies are detected using manifests and vulnerability databases, appropriate responses to alerts, the risks of ignoring alerts, developer responsibilities for alerts, access management for viewing alerts, and the placement of Dependabot alerts in the development process.
Topic 4	• Configure and use Code Scanning with CodeQL: This domain measures skills of Application Security Analysts and DevSecOps Engineers in code scanning using both CodeQL and third-party tools. It covers enabling code scanning, the role of code scanning in the development lifecycle, differences between enabling CodeQL versus third-party analysis, implementing CodeQL in GitHub Actions workflows versus other CI tools, uploading SARIF results, configuring workflow frequency and triggering events, editing workflow templates for active repositories, viewing CodeQL scan results, troubleshooting workflow failures and customizing configurations, analyzing data flows through code, interpreting code scanning alerts with linked documentation, deciding when to dismiss alerts, understanding CodeQL limitations related to compilation and language support, and defining SARIF categories.
Topic 5	• Configure and use secret scanning: This domain targets DevOps Engineers and Security Analysts with the skills to configure and manage secret scanning. It includes understanding what secret scanning is and its push protection capability to prevent secret leaks. Candidates differentiate secret scanning availability in public versus private repositories, enable scanning in private repos, and learn how to respond appropriately to alerts. The domain covers alert generation criteria for secrets, user role-based alert visibility and notification, customizing default scanning behavior, assigning alert recipients beyond admins, excluding files from scans, and enabling custom secret scanning within repositories.

Microsoft GitHub Advanced Security Sample Questions (Q21-Q26):

NEW QUESTION # 21
Which of the following information can be found in a repository's Security tab?

- A. GHAS settings
- B. Two-factor authentication (2FA) options
- C. Number of alerts per GHAS feature
- D. Access management

Answer: C

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.

NEW QUESTION # 22
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. Write access to the dependency manifest and lock files for an enterprise
- C. Dependency graph enabled at the organization level for all new private repositories
- D. Read-only access to the dependency manifest and lock files for a repository

Answer: C,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.

NEW QUESTION # 23
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 for Dependabot alerts
- B. enable all in existing repositories
- C. enable all for Dependency graph
- D. enable by default for new public repositories

Answer: A

Explanation:
To ensure you're notified whenever a vulnerability is detected via Dependabot, you mustenablealerts for Dependabotin your personal notification settings. This applies to both new and existing repositories. It ensures you get timely alerts about security vulnerabilities.
[Not C] The dependency graph must be enabled for scanning, but does not send alerts itself.

NEW QUESTION # 24
A repository's dependency graph includes:

- A. a summary of the dependencies used in your organization's repositories.
- B. dependencies from all your repositories.
- C. dependencies parsed from a repository's manifest and lock files.
- D. annotated code scanning alerts from your repository's dependencies.

Answer: C

Explanation:
The dependency graph includes all the dependencies of a repository that are detailed in the manifest and lock files, or their equivalent, for supported ecosystems, as well as any dependencies that are submitted using the dependency submission API. This includes:
Direct dependencies, that are explicitly defined in a manifest or lock file or have been submitted using the dependency submission API.

Indirect dependencies of these direct dependencies, also known as transitive dependencies or sub-dependencies.

NEW QUESTION # 25
Where can you find a deleted line of code that contained a secret value?

- A. Dependency graph
- B. Insights
- C. Commits
- D. Issues

Answer: A

Explanation:
Deleted lines of code containing secrets in a GitHub repository can still be accessed through the dependency graph and other tools, even after deletion. The dependency graph analyzes package manifest files to identify dependencies, including those in deleted or private repositories. Anyone with access to the dependency graph can potentially view the list of dependencies and their transitive dependencies, potentially exposing leaked secrets if they were previously part of the codebase.

NEW QUESTION # 26
.....