Prep4King's Microsoft GH-500 Practice Test Software (Web-Based and Desktop)

P.S. Free & New GH-500 dumps are available on Google Drive shared by Prep4King: https://drive.google.com/open?id=15_xOGlgjZ9cNre_mVOuAg9UACqug1gwX

We can provide you with a safety and efficiency shopping experience when you choose Prep4King GH-500 test Camp Questions. You see, we use Paypal to do the payment, so the payment process is secured and your personal information is secret and protected. In addition, the payment process is very easy to operate. You will receive an email attached with GH-500 study pdf after your payment in about 5-10 minutes, then you can start your study immediately.

Prep4King offers GitHub Advanced Security (GH-500) practice exams (desktop & web-based) which are customizable. It means candidates can set time and Microsoft GH-500 questions of the GitHub Advanced Security (GH-500) practice exam according to their learning needs. The Real GH-500 Exam environment of practice test help test takers to get awareness about the test pressure so that they become capable to counter this pressure during the final exam.

>> GH-500 Real Exams <<

Exam GH-500 Practice, Reliable GH-500 Test Answers

As our GitHub Advanced Security study questions can bring more professional quality service for the user. Our GH-500 study materials can give the user confidence and strongly rely on feeling, lets the user in the reference appendix not alone on the road, because we are to accompany the examinee on GH-500 Exam, candidates need to not only learning content of teaching, but also share his arduous difficult helper, so believe us, we are so professional company. Now, you can free download the demo of our GH-500 test guide to understand in more details.

Microsoft GH-500 Exam Syllabus Topics:

Topic	Details
Topic 1	• 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 2	• 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.
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	• 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 5	• 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.

Microsoft GitHub Advanced Security Sample Questions (Q24-Q29):

NEW QUESTION # 24
When configuring code scanning with CodeQL, what are your options for specifying additional queries? (Each answer presents part of the solution. Choose two.)

- A. github/codeql
- B. Queries
- C. Scope
- D. Packs

Answer: B,D

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.

NEW QUESTION # 25
Which of the following options would close a Dependabot alert?

- A. Viewing the dependency graph
- B. Viewing the Dependabot alert on the Dependabot alerts tab of your repository
- C. Leaving the repository in its current state
- D. Creating a pull request to resolve the vulnerability that will be approved and merged

Answer: D

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.

NEW QUESTION # 26
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 manifest and lock files
- C. In the dependency graph
- D. In security advisories reported on GitHub

Answer: A,D

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.

NEW QUESTION # 27
When using CodeQL, what extension stores query suite definitions?

- A. .yml
- B. .qls
- C. .qll
- D. .ql

Answer: B

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.

NEW QUESTION # 28
A repository's dependency graph includes:

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

Answer: C

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.

NEW QUESTION # 29
......

myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, www.stes.tyc.edu.tw, Disposable vapes

BTW, DOWNLOAD part of Prep4King GH-500 dumps from Cloud Storage: https://drive.google.com/open?id=15_xOGlgjZ9cNre_mVOuAg9UACqug1gwX