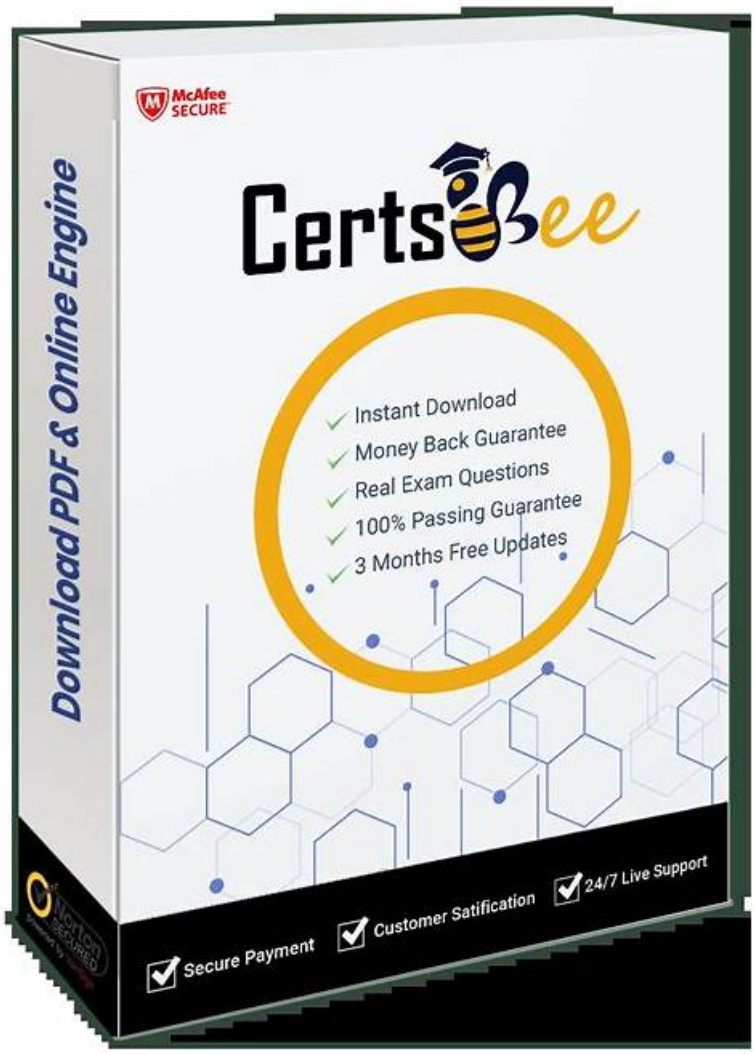
Relevant GH-500 Exam Dumps & Reliable GH-500 Dumps Free



BONUS!!! Download part of PDFBraindumps GH-500 dumps for free: https://drive.google.com/open?id=1DU0H2CxKTr76atSyleVvDyVnECAmQ8JU

The Microsoft GH-500 online exam is the best way to prepare for the Microsoft GH-500 exam. PDFBraindumps has a huge selection of GH-500 dumps and topics that you can choose from. The GH-500 Exam Questions are categorized into specific areas, letting you focus on the Microsoft GH-500 subject areas you need to work on.

If you have the GH-500 certification, it will be very easy for you to achieve your dream. But it is not an easy thing for many candidates to pass the GH-500 exam. By chance, our company can help you solve the problem and get your certification, because our company has compiled the GH-500 question torrent that not only have high quality but also have high pass rate. We believe that our GH-500 exam questions will help you get the certification in the shortest. So hurry to buy our GH-500 exam torrent, you will like our products.

>> Relevant GH-500 Exam Dumps <<

Reliable GH-500 Dumps Free | GH-500 Exam Pattern

We can assure you that you will get the latest version of our GH-500 training materials for free from our company in the whole year after payment. For we promise to give all of our customers one year free updates of our GH-500 exam questions and we update our GH-500 Study Guide fast and constantly. Do not miss the opportunity to buy the best GH-500 preparation questions in the

international market which will also help you to advance with the times.

Microsoft GH-500 Exam Syllabus Topics:

Topic	Details
Topic 1	• 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 2	• 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 3	• 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 4	• 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 5	• 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.

Microsoft GitHub Advanced Security Sample Questions (Q100-Q105):

NEW QUESTION # 100
After defining a secret scanning custom pattern, what is the final step before publishing the pattern?

- A. enabling push protection
- B. adding additional match requirements
- C. performing a dry run
- D. defining a custom pattern

Answer: C

Explanation:
Defining a custom pattern for a repository [See step 6 below]
Before defining a custom pattern, you must ensure that Secret Protection is enabled on your repository.
1. On GitHub, navigate to the main page of the repository.
2. Under your repository name, click Settings. If you cannot see the "Settings" tab, select the dropdown menu, then click Settings.
3. In the "Security" section of the sidebar, click Advanced Security.
4. Under "Secret Protection", to the right of "Custom patterns", click New pattern.
5. Enter the details for your new custom pattern. You must at least provide the name for your pattern, and a regular expression for the format of your secret pattern.
*-> 6. When you're ready to test your new custom pattern, to identify matches in the repository without creating alerts, click Save and dry run.
7. When the dry run finishes, you'll see a sample of results (up to 1000). Review the results and identify any false positive results.
8. Edit the new custom pattern to fix any problems with the results, then, to test your changes, click Save and dry run.
9. When you're satisfied with your new custom pattern, click Publish pattern.
10. Optionally, to enable push protection for your custom pattern, click Enable.

NEW QUESTION # 101
Where can you find the vulnerable dependencies that GitHub detected in your repository?

- A. in security advisories
- B. in code scanning alerts
- C. in secret scanning alerts
- D. in Dependabot alerts

Answer: D

Explanation:
Identifying vulnerabilities in your project's dependencies with Dependabot alerts Dependabot generates Dependabot alerts when known vulnerabilities are detected in dependencies that your project uses.
Dependabot alerts tab
If GitHub discovers insecure dependencies in your project, you can view details on the Dependabot alerts tab of your repository. Then, you can update your project to resolve or dismiss the alert.

NEW QUESTION # 102
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 enabled by default for new custom patterns.
- C. Push protection is not available for custom patterns.
- D. Push protection is an opt-in experience for each custom pattern.

Answer: D

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.

NEW QUESTION # 103
What classification is used to categorize Dependabot alerts? Each correct answer presents part of the solution. (Choose three.)

- A. Common Weakness Enumeration (CWE)
- B. Exploit Prediction Scoring System (EPSS)
- C. GitHub Security Advisory ID (GHSA)
- D. Static Application Security Testing (SAST)
- E. Common Vulnerabilities and Exposures (CVE)

Answer: A,B,E

Explanation:
[CE]
For enterprise organizations, GitHub's auto-triage rules help provide consistent management of security alerts at scale across multiple teams and repositories.
Auto-triage rules allow you to create custom criteria for automatically handling alerts based on factors like severity, EPSS [C], scope, package name, CVE[E], ecosystem, and manifest location.
You can create your own custom rules to control how Dependabot auto-dismisses and reopens alerts, so you can focus on the alerts that matter.
[D]
Common Weakness Enumeration (CWE) is used by CodeQL to describe the vulnerabilities it detects in code scanning alerts. CodeQL's queries are designed to identify a wide range of weaknesses, and each security query is associated with one or more specific CWEs, providing developers with standardized identifiers for the types of vulnerabilities found.
By associating alerts with CWEs, CodeQL provides a structured and informative approach to vulnerability management, making it easier for development teams to understand, address, and prevent security issues.
Note: The Common Weakness Enumeration (CWE) system is an industry-standard way of cataloging insecure software development patterns. CodeQL runs hundreds of queries out of the box that are able to detect an even greater number of CWEs. We went back through our existing queries, and aligned dozens of them with updated CWE IDs to give users better insight into the potential impact of a security issue when an alert is flagged up by code scanning.
Incorrect:
[Not A]
GitHub Advisories (GHSA) is a database of CVEs and GitHub-originated security advisories affecting the open source world. Advisories may or may not be documented in the National Vulnerability Database. Dependency-Track integrates with GHSA by mirroring advisories via GitHub's public GraphQL API.

NEW QUESTION # 104
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 security advisories reported on GitHub
- B. In the National Vulnerability Database
- C. In manifest and lock files
- D. In the dependency graph

Answer: A,B

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 # 105
.....