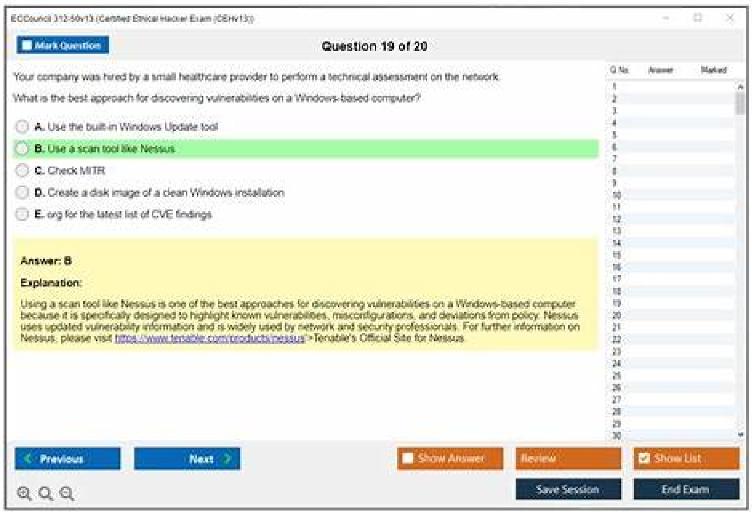
Free PDF Quiz Updated ECCouncil - Valid Test 312-97 Testking



Our 312-97 Exam Braindumps are the hard-won fruit of our experts with their unswerving efforts in designing products and choosing test questions. Pass rate is what we care for preparing for an examination, which is the final goal of our 312-97 certification guide. According to the feedback of our users, we have the pass rate of 99%, which is equal to 100% in some sense. The high quality of our products also embodies in its short-time learning. You are only supposed to practice EC-Council Certified DevSecOps Engineer (ECDE) guide torrent for about 20 to 30 hours before you are fully equipped to take part in the examination.

ECCouncil 312-97 Exam Syllabus Topics:

Topic	Details
Topic 1	• DevSecOps Pipeline - Build and Test Stage: This module explores integrating automated security testing into build and testing processes through CI pipelines. It covers SAST and DAST approaches to identify and address vulnerabilities early in development.
Topic 2	• DevSecOps Pipeline - Plan Stage: This module covers the planning phase, emphasizing security requirement identification and threat modeling. It highlights cross-functional collaboration between development, security, and operations teams to ensure alignment with security goals.
Topic 3	• DevSecOps Pipeline - Code Stage: This module discusses secure coding practices and security integration within the development process and IDE. Developers learn to write secure code using static code analysis tools and industry-standard secure coding guidelines.
Topic 4	• DevSecOps Pipeline - Release and Deploy Stage: This module explains maintaining security during release and deployment through secure techniques and infrastructure as code security. It covers container security tools, release management, and secure configuration practices for production transitions.
Topic 5	• Understanding DevOps Culture: This module introduces DevOps principles, covering cultural and technical foundations that emphasize collaboration between development and operations teams. It addresses automation, CI • CD practices, continuous improvement, and the essential communication patterns needed for faster, reliable software delivery.
Topic 6	• Introduction to DevSecOps: This module covers foundational DevSecOps concepts, focusing on integrating security into the DevOps lifecycle through automated, collaborative approaches. It introduces key components, tools, and practices while discussing adoption benefits, implementation challenges, and strategies for establishing a security-first culture.

>> Valid Test 312-97 Testking <<

Valid Test 312-97 Testking - Free PDF First-grade 312-97 - Practice EC-Council Certified DevSecOps Engineer (ECDE) Online

ECCouncil 312-97 exam dumps are important because they show you where you stand. After learning everything related to the EC-Council Certified DevSecOps Engineer (ECDE) (312-97) certification, it is the right time to take a self-test and check whether you can clear the 312-97 certification exam or not. People who score well on the 312-97 Practice Questions are ready to give the final EC-Council Certified DevSecOps Engineer (ECDE) (312-97) exam. On the other hand, those who do not score well can again try reading all the 312-97 dumps questions and then give the 312-97 exam.

ECCouncil EC-Council Certified DevSecOps Engineer (ECDE) Sample Questions (Q31-Q36):

NEW QUESTION # 31
(Jason Barry has been working as a DevSecOps engineer in an IT company that develops software products and applications for ecommerce companies. During the build-time check, Jason discovered SQL injection and XXS security issues in the application code. What action does the build-time check perform on the application code?.)

- A. It will send a message to issue and project management tool and continue with deploy-time check.
- B. It will stop the build process.
- C. It will send an alert to SIEM and continue with test-time check.
- D. It will ignore the security issue and continue the build process.

Answer: B

Explanation:
Build-time checks are designed to enforce security gates within the CI/CD pipeline. When critical vulnerabilities such as SQL injection and cross-site scripting (XSS) are detected during this stage, the correct and expected behavior is tofail the build. Stopping the build process prevents insecure code from progressing to later stages such as testing, deployment, or production. Ignoring issues or merely sending alerts while continuing the pipeline undermines the purpose of shift-left security. Alerts to SIEM systems and issue trackers are typically supplementary actions, but the primary enforcement mechanism at build time is to block the pipeline when severity thresholds are exceeded. This approach reduces remediation costs, limits exposure, and ensures that only secure artifacts move forward in the DevSecOps lifecycle.

NEW QUESTION # 32
(Jordon Garrett is working as a DevSecOps engineer in an IT company situated in Chicago, Illinois. His team prefers to use PowerShell for utilizing Git hooks because Bash and Windows are not compatible for advanced executions. For calling PowerShell script from Bash shell, Jordon wrote a PowerShell script using pre-commit logic such as pre-commit.ps1 and then executed the following commands
#!C:/Program\ Files/Git/usr/bin/sh.exe
exec powershell.exe -NoProfile -ExecutionPolicy Bypass -File "..git\hooks\pre-commit.ps1" How would Jordon know that the commit is successful?.)

- A. If the code exits with 0, then the commit is successful.
- B. If the code exits with 1, then the commit is successful.
- C. If the code exits with 2, then the commit is successful.
- D. If the code exits with 3, then the commit is successful.

Answer: A

Explanation:
Git hooks determine success or failure based on the exit code of the executed script. An exit code of0indicates successful execution, while any non-zero value signals failure and causes Git to abort the commit. In Jordon's setup, a Bash shell calls a PowerShell script to perform pre-commit checks. If the PowerShell script exits with code 0, Git interprets this as a successful hook execution and allows the commit to proceed. Exit codes such as 1, 2, or 3 indicate errors or policy violations and will block the commit. This mechanism ensures that security or quality checks enforced by the pre-commit hook must pass before code is committed. Using exit

codes in this way is a standard and reliable approach in cross-platform DevSecOps automation during the Code stage.

NEW QUESTION # 33
(Christopher Brown has been working as a DevSecOps engineer in an IT company that develops software and web applications for an ecommerce company. To automatically detect common security issues and coding error in the C++ code, she performed code scanning using CodeQL in GitHub. Which of the following entries will Christopher find for CodeQL analysis of C++ code?)

- A. CodeQL/Analyze (cpp) (pull-request).
- B. CodeQL/Analyze (cp) (pull-request).
- C. CodeQL/Analyze (cp) (push-request).
- D. CodeQL/Analyze (cpp) (push-request).

Answer: A

Explanation:
When GitHub Code Scanning is enabled using CodeQL, each supported programming language is identified by a specific language key. For C++ code, CodeQL uses the identifiercpp, not "cp." CodeQL workflows are commonly configured to run during pull request events so that security issues and coding errors can be detected and reviewed before code is merged into the main branch. As a result, the CodeQL analysis entry displayed in GitHub Actions and the Security tab for C++ pull request analysis appears asCodeQL/Analyze (cpp) (pull-request). Options A and B are incorrect because "cp" is not a valid CodeQL language identifier. Option C uses the correct language identifier but references an incorrect event format. Identifying the correct CodeQL analysis entry helps DevSecOps engineers confirm that scans are executing correctly for the intended language during the Code stage and that security feedback is available early in the development lifecycle.

NEW QUESTION # 34
(Judi Dench has recently joined an IT company as a DevSecOps engineer. Her organization develops software products and web applications related to electrical engineering. Judi would like to use Anchore tool for container vulnerability scanning and Software Bill of Materials (SBOM) generation. Using Anchore grype, she would like to scan the container images and file systems for known vulnerabilities, and would like to find vulnerabilities in major operating system packages such as Alpine, CentOS, Ubuntu, etc. as well as language specific packages such as Ruby, Java, etc. Which of the following commands should Judi run to scan for vulnerabilities in the image using grype?)

- A. grype packages < image > --scope all-layers.
- B. grype < image > --scope all-layers.
- C. grype packages < image >.
- D. grype < image >.

Answer: B

Explanation:
Grype is a vulnerability scanning tool used to analyze container images and file systems for known vulnerabilities across operating system and application dependencies. The most effective way to perform a comprehensive scan is by running the grype <image> --scope all-layers command. This ensures that vulnerabilities are detected acrossall layersof the container image, not just the final runtime layer. Containers often inherit vulnerabilities from base images or intermediate layers, making full-layer scanning essential. The packages subcommand is used for listing detected packages rather than performing vulnerability analysis.
Running Grype during the Build and Test stage allows DevSecOps teams to identify vulnerable base images and dependencies early, reducing the risk of deploying insecure containers into production and supporting secure container lifecycle management.

NEW QUESTION # 35
(Alexander Hamilton has been working as a senior DevSecOps engineer in an IT company located in Greenville, South Carolina. In January of 2012, his organization because a victim of a cyber security attack and incurred a tremendous loss. Alexander's organization immediately adopted AWS cloud-based services after the attack to develop robust software products securely and quickly. To detect security issues in code review, Alexander would like to integrate SonarQube with AWS Pipeline; therefore, he created a pipeline in AWS using CloudFormation pipeline template. Then, he selected SonarQube tool from the tools dropdown,

provided the required stack parameters, and also provided email address for receiving email notifications of changes in pipeline status and approvals. He deployed the pipeline after entering the required information.
What will happen when changes are committed in the application repository?.)

- A. BinSkim event is created.
- B. CloudWatch event is created.
- C. Cloud Config event is created.
- D. Security Hub event is created.

Answer: B

Explanation:
When changes are committed to a repository connected to an AWS Pipeline, the pipeline execution is triggered and monitored usingAmazon CloudWatch events. CloudWatch captures pipeline state changes, execution status, and approval notifications, enabling real-time monitoring and alerting. AWS Config tracks resource configuration changes, BinSkim is a binary analysis tool, and Security Hub aggregates security findings but does not directly track pipeline execution events. Integrating SonarQube into AWS Pipeline ensures static code analysis runs automatically upon commits, while CloudWatch provides visibility into pipeline activity. This setup strengthens security automation during the Code stage by ensuring every commit is analyzed and monitored.

NEW QUESTION # 36
......