PassQueen
a.k.a dumpsqueen.com

EC-Council Certified DevSecOps Engineer (ECDE)

ECCouncil 312-97

Version Demo

Total Demo Questions: 13

Total Premium Questions: 133

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Understanding DevOps Culture	2
Topic 2, Designing and Implementing DevSecOps Pipeline	5
Topic 3, Building a Secure Continuous Integration (CI) Pipeline	50
Topic 4, Building a Secure Continuous Delivery (CD) Pipeline	3
Topic 5, Implementing Continuous Feedback	6
Topic 6, Implementing Infrastructure as Code (IaC)	8
Topic 7, Implementing Container Security	19
Topic 8, Implementing Application Security	36
Topic 9, Implementing Security in Kubernetes	4
Total	133

QUESTION NO: 1

(Charlotte Flair is a DevSecOps engineer at Egma Soft Solution Pvt. Ltd. Her organization develops software and applications related to supply chain management. Charlotte would like to integrate Sqreen RASP tool with Slack to monitor the application at runtime for malicious activities and block them before they can damage the application. Therefore, she created a Sqreen account and installed Sqreen Microagent. Now, she would like to install the PHP microagent. To do so, she reviewed the PHP microagent's compatibility, then she signed in to Sqreen account and noted the token in Notepad. Which of the following commands should Charlotte run in the terminal to install the PHP extension and the Sqreen daemon?.)

A. curl -s https://download.sqreen.com/php/install.sh > sqreen-install.sh \ && bash sqreen-install.sh [CHARLOTTE'S ORG TOKEN HERE] "[CHARLOTTE'S APP NAME HERE]".

B. curl -shttps://download.sqreen.com/php/install.sh < sqreen-install.sh \ && bash sqreen-install.sh [CHARLOTTE'S ORG TOKEN HERE] "[CHARLOTTE'S APP NAME HERE]".

C. curl -ihttps://download.sqreen.com/php/install.sh > sqreen-install.sh \ && bash sqreen-install.sh [CHARLOTTE'S ORG TOKEN HERE] "[CHARLOTTE'S APP NAME HERE]".

D. curl -ihttps://download.sqreen.com/php/install.sh < sqreen-install.sh \ && bash sqreen-install.sh [CHARLOTTE'S ORG TOKEN HERE] "[CHARLOTTE'S APP NAME HERE]".

ANSWER: A

Explanation:

The command `curl -s https://download.sqreen.com/php/install.sh > sqreen-install.sh && bash sqreen-install.sh [CHARLOTTE'S ORG TOKEN HERE] "[CHARLOTTE'S APP NAME HERE]"` is the appropriate installation pattern because it first retrieves the Sqreen PHP installer script and writes the downloaded content to the local file `sqreen-install.sh`. The `-s` flag runs curl in silent mode, which is suitable for an automated installer download. The shell's `&&` operator ensures that the installer is invoked only after the download command completes successfully. Running the script with the organization token associates the installed agent with Charlotte's Sqreen organization, while the application-name argument identifies the protected PHP application in the Sqreen console. The installer is intended to set up both the PHP extension used to instrument PHP requests and the local Sqreen daemon/microagent component required for runtime protection. In a production DevSecOps workflow, the organization token should be supplied through a protected secret-management mechanism rather than embedded in source code or shell history. Sqreen's historical PHP agent source and installation-related artifacts are available in the Sqreen PHP Agent repository; Sqreen is now part of Datadog.

QUESTION NO: 2

(John Cho has been working as a senior DevSecOps engineer in an IT company that develops Node.js web applications. The team uses Retire.js in Jenkins to identify vulnerable JavaScript libraries. A scan reports that a vulnerable library is present only in a development dependency used for local test tooling and is not included in the production package. Which of the following is the most appropriate next action?)

A. Ignore the alert because all development dependencies are harmless.

B. Verify production reachability and evaluate the finding according to dependency scope and policy.

C. Disable Retire.js scanning for all future Jenkins builds.

D. Promote the artifact without reviewing the dependency tree.

ANSWER: B

Explanation:

The team should verify the dependency's production reachability and apply the organization's policy based on the actual release risk. A development-only dependency may not affect the deployed runtime artifact, but it can still affect build agents, developer systems, or pipeline integrity. The finding should therefore be triaged with dependency scope and build usage considered rather than automatically treated as identical to a production runtime vulnerability.

Ignoring the alert without review is unsafe. Promoting the artifact without determining whether the dependency is packaged or executed does not provide a defensible decision. Disabling all Retire.js checks removes useful software-composition visibility.

QUESTION NO: 3

(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. Alexanderâ€™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. Cloud Config event is created.

B. BinSkim event is created.

C. CloudWatch event is created.

D. Security Hub event is created.

ANSWER: C

Explanation:

CloudWatch event is created. A commit to the repository configured as the pipeline's source initiates a new pipeline execution. AWS CodePipeline publishes events for pipeline, stage, action, and approval state changes to Amazon EventBridge; Amazon EventBridge is the current name for the service formerly called Amazon CloudWatch Events. Thus, in course material or templates using the older terminology, this behavior is described as creation of a CloudWatch event. The event provides a structured notification that the pipeline execution or one of its states has changed, and it can be routed to targets such as Amazon SNS for the configured email notifications, AWS Lambda, or other automation. The SonarQube integration is then invoked in the relevant pipeline stage so the newly committed application code can undergo the configured static-analysis process. This event-driven model allows pipeline activity, including execution progress and approvals, to be monitored and acted upon without polling. AWS documents CodePipeline state-change events under its Amazon EventBridge integration, including events generated when pipeline executions change state. See AWS CodePipeline: monitoring state changes with Amazon EventBridge and Amazon EventBridge User Guide.

QUESTION NO: 4

(Trevor Noah has been working as a DevSecOps engineer in an IT company located in Detroit, Michigan. His team leader asked him to perform continuous threat modeling using ThreatSpec. To do so, Trevor installed and initialized ThreatSpec in the source code repository; he then started annotating the source code with security issues, actions, or concept. Trevor ran ThreatSpec against the application code and he wants to generate the threat model report. Which of the following command Trevor should use to generate the threat model report using ThreatSpec?.)

A. $ ThreatSpec report.

B. $ ThreatSpec Report.

C. $ Threatspec Report.

D. $ threatspec report.

ANSWER: D

Explanation:

`threatspec report` is the correct command because ThreatSpec uses the lowercase executable name `threatspec` and provides `report` as the subcommand for producing a threat-model report from annotations discovered in the codebase. Once developers have initialized the repository and embedded ThreatSpec annotations representing security concepts, threats, mitigations, and related relationships, this command processes those annotations into a documented model that can be reviewed and maintained alongside the application. The command-line syntax is case-sensitive in typical development environments, so the executable and subcommand should be entered exactly as documented. Running report generation as part of the development workflow supports continuous threat modeling: changes to source code and its annotations can be reflected in an updated threat model rather than relying on a one-time design exercise. This creates a practical feedback loop for DevSecOps teams by making security assumptions, identified threats, and planned controls visible in an artifact that can be reviewed during engineering work. ThreatSpec's project materials describe its code-annotation approach and reporting capability; see the ThreatSpec GitHub repository and the ThreatSpec project site.

QUESTION NO: 5

(Terry Diab has been working as a DevSecOps engineer in an IT company that develops software products and web applications for a call center. She would like to integrate Snyk with AWS CodeCommit to monitor and remediate vulnerabilities in the code repository. Terry pushed code to AWS CodeCommit; this triggered Amazon EventBridge Rule, which then triggered AWS CodePipeline. AWS CodePipeline passed code to Snyk CLI run. Who among the following interacts with Snyk CLI and sends the results to Snyk UI?)

A. AWS CodeDeploy.

B. AWS CodeCommit.

C. AWS Pipeline.

D. AWS CodeBuild.

ANSWER: D

Explanation:

AWS CodeBuild. is the correct component because it provides the managed build environment in which the Snyk CLI command is executed. CodePipeline coordinates the overall CI/CD workflow and passes source artifacts between stages, but a build action uses CodeBuild to run the commands defined in a build specification. In this design, CodeBuild retrieves the source artifact supplied by the pipeline and runs commands such as `snyk test` or `snyk monitor`. The Snyk CLI authenticates to Snyk using an organization token configured as a protected environment variable or retrieved securely at build time. Once invoked, the CLI analyzes project dependencies and communicates directly with Snyk's service to publish monitoring or test results, which are then available in the Snyk UI. This arrangement enables vulnerability checks to run automatically on each relevant repository change while retaining pipeline orchestration and build execution as separate responsibilities. AWS documents that CodeBuild executes build commands from a buildspec file, including commands for third-party tooling. Snyk also documents its CLI commands and authentication workflow for uploading monitored project snapshots. See AWS CodeBuild buildspec reference and Snyk CLI monitor command documentation.

QUESTION NO: 6

(Brady Coleman is a senior DevSecOps engineer at CloudVac Security Private Ltd. He has created a new container named "eccbrad" from the centos:7 image using the command docker run -i -t --name geeklab centos:7 /bin/bash. Now, Brady wants to install the httpd package inside the eccbrad container. Which of the following commands should Brady use to install the httpd package inside the container?)

A. sudo install-httpd.

B. sudo install httpd.

C. yum install-httpd.

D. yum install httpd.

ANSWER: D

Explanation:

`yum install httpd.` is the appropriate command for installing the Apache HTTP Server package in a container based on the `centos:7` image. CentOS 7 uses YUM as its package manager, and the HTTP Server package is named `httpd`. Once the interactive container shell is running, executing `yum install httpd` resolves the package from enabled repositories, downloads it together with any required dependencies, and installs it into the container filesystem. In practical use, the command is commonly written as `yum install -y httpd` in a Dockerfile or noninteractive automation so that installation does not pause for confirmation. The question's command invokes an interactive Bash shell, so the package command is intended to be run from that shell. Container images generally run processes as the root user by default unless a different `USER` is configured; therefore, `sudo` is normally unnecessary and may not even be installed in a minimal image. After installation, Apache can be launched with `httpd` or an appropriate foreground command, depending on the container's intended runtime configuration. Red Hat documents YUM package installation in its YUM guidance, and Apache identifies `httpd` as its server software in the Apache HTTP Server documentation.

QUESTION NO: 7

(Rahul Mehta is working as a DevSecOps engineer in an IT company that develops cloud-native web applications. His organization follows a strict DevSecOps practice and wants to ensure that third-party open-source dependencies used in the application do not introduce known security vulnerabilities. Rahul decided to integrate a Software Composition Analysis (SCA) tool into the CI pipeline so that every build is automatically scanned. During one of the builds, the SCA tool detects a critical vulnerability in a transitive dependency. What should ideally happen in a mature DevSecOps pipeline when such a critical vulnerability is detected at build time?.)

A. The pipeline should log the vulnerability details and continue the build to avoid delivery delays.

B. The pipeline should notify the security team and continue with deploy-time checks.

C. The pipeline should fail the build and prevent the artifact from progressing further.

D. The pipeline should ignore transitive dependencies and only scan direct dependencies.

ANSWER: C

Explanation:

The pipeline should fail the build and prevent the artifact from progressing further. A mature DevSecOps implementation treats critical findings from Software Composition Analysis as an enforceable policy gate in continuous integration. The purpose of scanning dependencies during the build is to identify known vulnerable components early, before the resulting artifact is promoted to testing, release, or production environments. This includes transitive dependencies, because they are incorporated into the application's dependency graph and can expose the application even when developers did not add them directly.

Failing the build creates an immediate, traceable remediation workflow. The team can update the affected package, upgrade the parent dependency that introduced it, select a secure replacement, or apply an approved mitigation and exception process when no timely fix exists. Crucially, the artifact should not advance unless it meets the organization's defined vulnerability threshold. This embodies the shift-left principle by making security feedback actionable at the point where dependency changes are introduced, while preventing known-critical risk from being carried forward into later delivery stages. NIST's Secure Software Development Framework recommends identifying and addressing vulnerabilities in third-party software components, and OWASP describes dependency analysis as a core practice for detecting publicly disclosed vulnerable components. See NIST SP 800-218 SSDF and OWASP Dependency-Check.

QUESTION NO: 8

(Craig Kelly has been working as a software development team leader in an IT company over the past 8 years. His team is working on the development of an Android application product. Sandra Oliver, a DevSecOps engineer, used DAST tools and fuzz testing to perform advanced checks on the Android application product and detected critical and high severity issues. She provided the information about the security issues and the recommendations to mitigate them to Craigâ€™s team. Which type of security checks performed by Sandra involve detection of critical and high severity issues using DAST tools and fuzz testing?)

A. Commit-time checks.

B. Build-time checks.

C. Deploy-time checks.

D. Test-time checks.

ANSWER: D

Explanation:

Test-time checks. are the appropriate category because Dynamic Application Security Testing (DAST) and fuzz testing assess an application while it is running, typically in an isolated test or staging environment. DAST sends requests to the deployed application and evaluates its runtime responses to identify exploitable behaviors and weaknesses in areas such as input handling, authentication, session management, and error processing. Fuzzing similarly supplies unexpected, malformed, or randomized inputs to an executing application or its interfaces to expose crashes, unhandled exceptions, and security-relevant flaws. These techniques therefore require a functional application build that can be exercised dynamically rather than merely inspected as source code or an artifact. In a DevSecOps pipeline, conducting these checks during testing enables the team to identify and prioritize critical and high-severity findings before release, then remediate and retest them as part of the delivery workflow. This approach aligns with OWASP guidance that DAST tests a running application externally, and with NIST's description of fuzz testing as automated input generation used to discover vulnerabilities in software. See OWASP: Dynamic Application Security Testing and NIST: Fuzz Testing.

QUESTION NO: 9

(Matt LeBlanc has been working as a DevSecOps engineer in an IT company that develops software products and web applications for IoT devices. His team leader has asked him to use GitRob tool to find sensitive data in the organizational public GitHub repository. To install GitRob, Matt ensured that he has correctly configured Go >= 1.8 environment and that $GOPATH/bin is in his $PATH. The GitHub repository URL from which he is supposed to install the tool ishttps://github.com/michenriksen/gitrob . Which of the following command should Matt use to install GitRob?.)

A. $ go get github.com/michenriksen/gitrob

B. $ go get gitrob github.com/michenriksen/gitrob.

C. $ go git github.com/michenriksen/gitrob.

D. $ go git gitrob github.com/michenriksen/gitrob.

ANSWER: A

Explanation:

`$ go get github.com/michenriksen/gitrob` is the correct installation command for the Go environment described in the question. In the pre-module Go workflow used by Go 1.8, `go get` accepts an import path, downloads the corresponding source repository and its dependencies into the configured GOPATH workspace, then builds and installs the executable. Because `$GOPATH/bin` is already included in `$PATH`, the resulting `gitrob` binary can be invoked directly from a terminal after installation. The import path supplied to `go get` is derived directly from the repository's GitHub location: `github.com/michenriksen/gitrob`. GitRob is intended to help identify potentially sensitive files and data exposed through GitHub repositories, making it relevant to a DevSecOps review of public organizational repositories. The project repository documents GitRob and its source location at GitHub: michenriksen/gitrob. Go's historical GOPATH-based package-installation behavior is documented in the Go GOPATH workspace documentation.

QUESTION NO: 10

(Teresa Wheeler is a DevSecOps engineer at Altschutz Solution Pvt. Ltd. She would like to test the web applications and APIâ€™s from outside without accessing the source code using BDD security framework. The framework is a collection of Cucumber-JVM features that are pre-configured with OWASP ZAP, Nessus scanner, SSLyze, and Selenium. Hence, she downloaded and ran the jar application, and then cloned the BDD security framework. Next, she utilized a command for

executing the authentication feature. Which of the following commands allows Teresa to execute all the features of BDD security framework, including the OWASP ZAP?.)

A. ./gardlew.

B. /gardlev.

C. /gardlew.

D. ./gardlev.

E. ./gradlew.

ANSWER: E

Explanation:

`./gradlew` is the correct command because it invokes the Gradle Wrapper script located in the root directory of the cloned BDD-Security project. The initial `./` is significant on Unix-like systems: it explicitly runs the local wrapper file from the current directory rather than relying on a system-wide Gradle installation. The wrapper downloads and uses the Gradle version specified by the project, making execution repeatable across development and CI/CD environments.

BDD-Security uses Gradle to orchestrate its Cucumber-JVM security features and the integrated security-testing tools, including OWASP ZAP. Invoking the project's wrapper without a misspelling or an incorrect absolute-path prefix allows its configured default build behavior to execute the framework's configured feature suite. This supports black-box, behavior-driven security validation of web applications and APIs, including the automated ZAP scan workflow. The project source and usage material are available in the BDD-Security GitHub repository. Gradle's documentation also explains that the Wrapper is run as `./gradlew` on Unix systems and is the recommended way to execute a project's build consistently: Gradle Wrapper documentation.

QUESTION NO: 11

(Brett Ryan has been working as a senior DevSecOps engineer in an IT company in Charleston, South Carolina. He is using git-mutimail tool to send email notification for every push to git repository. By default, the tool will send one output email providing details about the reference change and one output email for every new commit due to a reference change. How can Brett ensure that git-multimail is set up appropriately?)

A. Running the environmental variable GITHUB_MULTIMAIL_CHECK_SETUP by setting it to non-empty string.

B. Running the environmental variable GIT_MULTIMAIL_CHECK_SETUP by setting it to empty string.

C. Running the environmental variable GIT_MULTIMAIL_CHECK_SETUP by setting it to non-empty string.

D. Running the environmental variable GITHUB_MULTIMAIL_CHECK_SETUP by setting it to empty string.

ANSWER: C

Explanation:

Running the environmental variable GIT_MULTIMAIL_CHECK_SETUP by setting it to non-empty string. is correct because `git-multimail` provides the `GIT_MULTIMAIL_CHECK_SETUP` environment variable specifically to validate whether its hook and configuration are set up correctly. When this variable is present and has a non-empty value, the tool performs its setup check rather than following its normal notification-processing path. This gives an administrator a practical way to verify the installation and configuration before relying on the hook to distribute push and commit notifications. Such validation is valuable in a DevSecOps workflow because server-side Git hooks execute in a restricted environment that can differ from an interactive shell: required configuration, executable paths, permissions, and mail-delivery settings should be confirmed in the actual hook context. The upstream git-multimail documentation identifies `GIT_MULTIMAIL_CHECK_SETUP` as the environment variable used for this check and describes the expected setup behavior. See the project's official git-multimail repository and its README and configuration guidance for installation, hook placement, and validation details.

QUESTION NO: 12

(Dustin Hoffman is a DevSecOps engineer at SantSol Pvt. Ltd. His organization develops software products and web applications related to mobile apps. Using Gauntlt, Dustin would like to facilitate testing and communication between teams and create actionable tests that can be hooked in testing and deployment process. Which of the following commands should Dustin use to install Gauntlt?.)

A. $ gems install Gauntlt.

B. $ gems install gauntlt.

C. $ gem install gauntlt.

D. $ gem install Gauntlt.

ANSWER: C

Explanation:

`$ gem install gauntlt.` is the intended installation command. Gauntlt is distributed as a RubyGem, so it is installed through RubyGems' `gem install` command. In practical shell use, the trailing period is question punctuation rather than part of the command; the command entered at a terminal is `gem install gauntlt`. The gem name is lowercase, matching the package's published RubyGems identifier. Installing it this way makes the Gauntlt command-line tooling available for running security-oriented attack scenarios as executable tests in a delivery pipeline. This supports the described DevSecOps workflow because Gauntlt tests can be authored in a collaboration-friendly format and invoked repeatedly during build, test, or deployment automation. The project's installation guidance specifically identifies RubyGems installation with `gem install gauntlt`, and the RubyGems package listing confirms that `gauntlt` is the published gem name. See the Gauntlt GitHub repository and the Gauntlt RubyGems package page.

QUESTION NO: 13

(Teresa Wheeler is a DevSecOps engineer at Altschutz Solution Pvt. Ltd. She would like to test the web applications and API's from outside without accessing the source code using BDD security framework. The framework is a collection of Cucumber-JVM features that are pre-configured with OWASP ZAP, Nessus scanner, SSLyze, and Selenium. Hence, she downloaded and ran the jar application, and then cloned the BDD security framework. Next, she utilized a command for executing the authentication feature. Which of the following commands allows Teresa to execute all the features of BDD security framework, including the OWASP ZAP?.)

A. ./gradlew

B. /gardlev.

C. /gardlew.

D. ./gardlev.

ANSWER: A

Explanation:

`./gradlew` is the correct command because it invokes the Gradle Wrapper script contained in a cloned BDD-Security project on a Unix-like system. The leading `./` explicitly runs the wrapper located in the current working directory, rather than relying on a system-wide Gradle installation. The wrapper uses the Gradle version specified by the project, helping ensure that its Cucumber-JVM feature suite and configured security-testing tasks run with the dependencies and build settings expected by the framework. When invoked without a task name, Gradle runs the project's default task or tasks. In the BDD-Security framework, this provides the standard entry point for running the complete configured feature set, including the OWASP ZAP-based external web-application security tests. This aligns with the framework's purpose: executing behavior-driven security checks against a running application without requiring access to its source code. Gradle documents that the Wrapper is executed as `./gradlew` on Unix systems, and the BDD-Security project documents its Gradle-based execution workflow. See the Gradle Wrapper documentation and the BDD-Security project repository.