KALASALINGAM
ACADEMY OF RESEARCH AND EDUCATION
(DEEMED TO BE UNIVERSITY)
Under sec. 3 of UGC Act 1956.

Anand Nagar, Krishnankoil - 626126. Srivilliputtur (Via), Virudhunagar (Dt), Tamil Nadu | info@kalasalingam.ac.in | www.kalasalingam.ac.in

Mini Project Report

IT DATA SECURITY (213CSE4309)

Team Number	03
Project Title	Integration of Security in DevSecOps Pipelines
Team Members (Reg. No & Name)	Barathi M 99220041124
	M.Abitha -99220040794
	S.Remina-99220040712
	M.Nivetha -99220040666
Faculty Incharge	Dr K. Kartheeban

Table of Contents

ABSTRACT

In the modern software development era, integration of security into the Continuous Integration and Continuous Deployment pipeline is a must, in order to protect an application from vulnerabilities. In this project, the implementation of a DevSecOps pipeline for a web application with emphasis on its automated security practices throughout the entire software development cycle is aimed at ensuring the integrity and quality of the code.

Jenkins acts as the CI/CD automation server and triggers the pipeline whenever the GitHub repository identifies changes. Once the pipeline is initiated, it runs several stages that include the analysis of code quality using SonarQube, OWASP Dependency Check to scan for vulnerabilities, and filesystem security analysis with Trivy. It has thus been possible to detect possible vulnerabilities or code issues quite early in the lifecycle.

The final step is secure deployment via Docker Compose, striving to achieve an efficient and resilient release process. By including security practices into such a pipeline, this project manifests how DevSecOps can in fact improve code quality as well as application security in general.

INTRODUCTION:

Since cyber threats are increasingly on the rise, it is imperative that applications be secured as early as the beginning of their development. Under traditional software development, security usually comes in as an afterthought, with possible vulnerabilities left during and at the stages of production. DevSecOps does this by infusing security practices right into the CI/CD pipeline with shared responsibility from teams in development, security, and operations.

The pipeline of DevSecOps would then be built for the application hosted in GitHub, using Jenkins to automate the whole flow of CI/CD. The starting step would be cloning the code from GitHub, followed by conducting SonarQube scans for analyses about code quality and security checks. After this, OWASP Dependency Check is used to identify known vulnerabilities within third-party libraries, ensuring that all external dependencies are secured. The SonarQube quality gate will ensure that your code passes certain thresholds, and Trivy will then scan your filesystem for further threats. The deployment phase is safeguarded by using Docker Compose, which enables you to make deployments consistent and secure.

The best practices of DevSecOps ensure both code quality and quality security when developing software. This project demonstrates how checking is automated at every stage to avoid risks that might be associated with reaching production with vulnerabilities, thus evoking a proactive approach to application security.

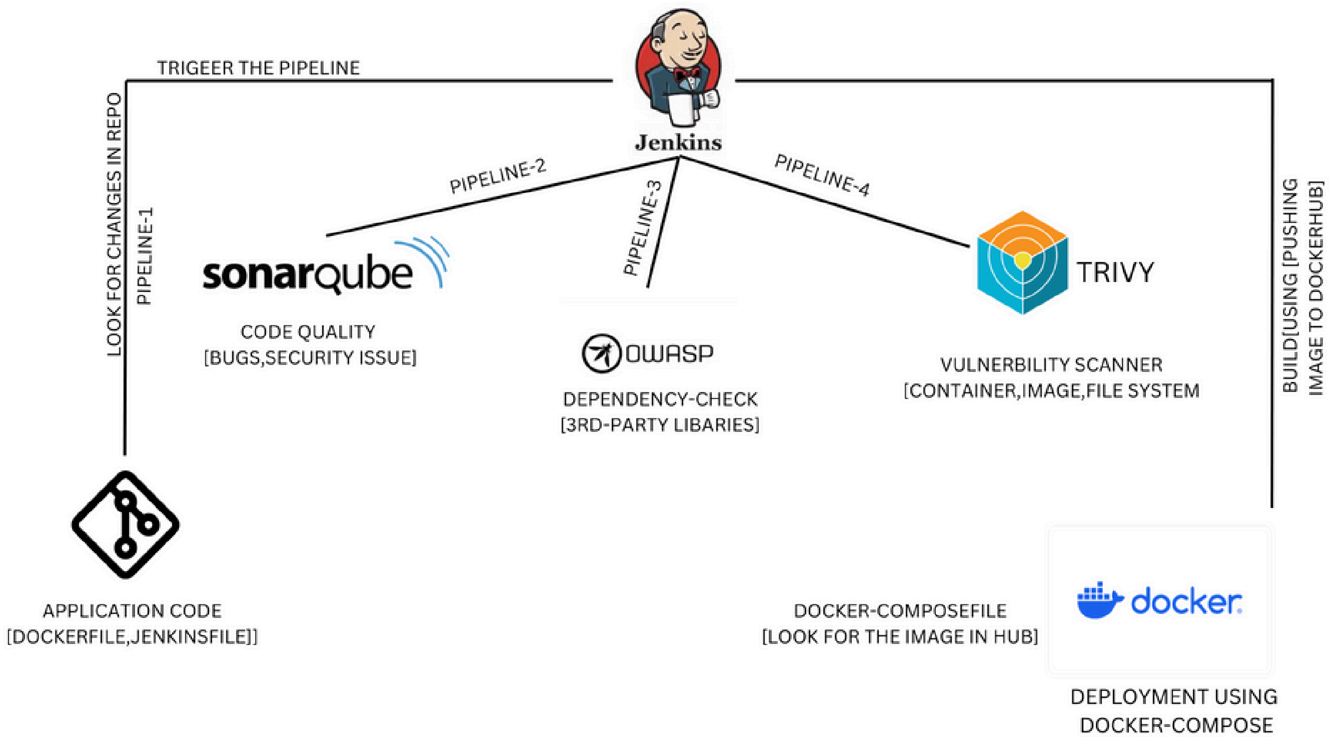
OBJECTIVE

This project will execute a DevSecOps pipeline for a web application with security practices augmented in the CI/CD process. The intent is the automation of security checks throughout the development lifecycle so that vulnerabilities are caught and addressed before deployment.

Jenkins and SonarQube, OWASP Dependency Check, Trivy, and Docker Compose are the tools implemented in the pipeline for these purposes. Each tool makes its own particular contribution: from code quality analysis up to the vulnerability scanning of the component with the safe later deployment.

This ultimately leads to a pipeline that promotes the use of safe coding practices and yet ensures a good quality of code throughout the entire developmental process. The automation of these security measures by the project will ultimately result in reduced potential risks, which leads to a secure, robust application.

WORKFLOW:


TRIGEER THE PIPELINE
LOOK FOR CHANGES IN REPO
PIPELINE-1
Jenkins
PIPELINE-2
PIPELINE-3
PIPELINE-4
sonarqube
CODE QUALITY
[BUGS,SECURITY ISSUE]
OWASP
DEPENDENCY-CHECK
[3RD-PARTY LIBARIES]
TRIVY
VULNERBILITY SCANNER
[CONTAINER,IMAGE,FILE SYSTEM
BUILD[USING [PUSHING
IMAGE TO DOCKERHUB]
APPLICATION CODE
[DOCKERFILE,JENKINSFILE]]
DOCKER-COMPOSEFILE
[LOOK FOR THE IMAGE IN HUB]
docker
DEPLOYMENT USING
DOCKER-COMPOSE

Tools and Technologies Used

Jenkins:

Jenkins will be the automation server that will aid in achieving the CI/CD. It will automatically run the pipeline once any code changes are made in the GitHub repository. This means all the build, test, and deployment will be automated.

GitHub:

The codebase will be hosted and tracked on a version control system - GitHub. The application integrates with Jenkins to run the pipeline executions based on either the code or pull requests made.

SonarQube:

SonarQube is a static code analyzer that can discover code smells, bugs, and security vulnerabilities. It will make an in-depth review of the codebase to guarantee the quality of the software as well as find potential issues which may compromise security or maintainability.

OWASP Dependency Check:

This tool scans over the dependencies in your project and flags known vulnerabilities in third-party libraries. This ensures any external element of the project does not introduce security risks.

Trivy:

It scans filesystem security and detects vulnerabilities in system components, dependencies, and configurations. It secures all aspects of the runtime and dependencies for the application.

Docker Compose:

It is used for managing and deploying the application in a consistent and isolated environment. It further supports the deployment process by allowing automated deployment with the services necessary, going in increasing directions of supporting security and efficiency in the deployment process.

METHODOLOGY:

The methodology used in this DevSecOps project is by injecting security into the Continuous Integration and Continuous Deployment pipeline such that all secure practices are looked at in the lifecycle of software development. The implementation would follow the best practice for DevSecOps by automating the checks on security and quality from code commit to deployment. Below are some of the key components of the methodology used:

Automation of the CI/CD Pipeline (Jenkins)
Jenkins is actually the largest automation tool for the CI/CD process. It continually checks the GitHub repository for the codebase changes. Whenever a commit or a pull request gets detected, it initiates the pipeline. This way, the feedback can reach the development team quickly about the code changes, whereas the entire process of building, testing, and deployment remains automated. Jenkins ensures that all steps, right from code analysis to deployment, are processed in an integrated way, making the pipeline efficient and predictable.

Code Quality and Vulnerability Analysis (SonarQube)
Once the code is pulled from GitHub, SonarQube runs its static code analysis. It checks for bugs, smells in the code, and security issues. It identifies weaknesses in the code early on so that the developers can fix them and take the application ahead in the pipeline. The check includes any non-compliance with coding standards or best practices and know security flaws in the code.

OWASP Dependency Check: Dependency Scanning
OWASP Dependency Check is another scanning activity in the pipeline after the analysis by SonarQube. This tool is used to identify known vulnerabilities in libraries or frameworks that the application relies on. It ensures that, through the scanning of project dependencies, third-party packages or other components used for the application do not pose any form of security risk. Whenever the pipeline detects any vulnerable dependencies, it reports back to the development team so that they upgrade or replace the concerned vulnerable components accordingly.

Quality Gate Verification by SonarQube Quality Gate
The pipeline integrates a SonarQube quality gate check after static analysis. This ensures that the code is at pre-defined quality thresholds, such as acceptable levels of bugs, code smells, and vulnerabilities. If the code doesn't meet these criteria, it stops the pipeline, so that no further stages- like deployment-will continue. This step ensures only high-quality, sound, and secure code moves forward in the pipeline.

Filesystem Security Scanning- Trivy
Trivy is a filesystem scanning tool that identifies system dependencies and configurations' vulnerabilities. It scans container images, OS packages, and other used resources by an application to ensure that vulnerabilities that may arise from outdated or misconfigured parts of the system are tackled before deploying an application.

Secure Deployment-Docker Compose
Deploy the application using Docker Compose. Docker Compose orchestrates the services of your application to deploy in a consistent and secure manner. Docker containers create a kind of isolated environment that diminishes the risk of security vulnerabilities during the deployment process. Automating the deployment process, the pipeline ensures that the application is secured and launched efficiently in a production-like environment.

CONCLUSION

Software development lifecycle security is no longer an option but a must-have. This DevSecOps project shows how embedding security directly into the CI/CD pipeline not only enhances quality but also application security while spawning a collaborative culture with mutual responsibility. The proactive project, which mainly focuses on automating at every stage of development-from code analysis to deployment-through key security and quality checks, has helped bring out the strong power of proactive security measures in protecting applications against vulnerableness.

As we head toward a world in which software development will be faster, much more complex, and connected increasingly, the role of DevSecOps is paramount. The security automation and continuous feedback make it possible for the developers to produce secure resilient applications, even as they maintain the needed speed and agility. This project will really be capable of being transformed with DevSecOps: innovation meets security.

With the unfolding world of technology, this project teaches lessons as a base for building even more secure and efficient pipelines. "We're not just building software," he said, but trust and reliability-and a safer digital future for everyone-in the process.

REFERENCES

Jenkins Documentation
Jenkins, "Jenkins User Documentation," Jenkins Project, https://www.jenkins.io/doc/.

SonarQube Documentation
SonarQube, "SonarQube Documentation," SonarSource, https://docs.sonarqube.org/.

OWASP Dependency-Check
OWASP, "OWASP Dependency-Check," Open Web Application Security Project, https://owasp.org/www-project-dependency-check/.

Trivy Documentation
Aqua Security, "Trivy: A Simple and Comprehensive Vulnerability Scanner for Containers and other Artifacts," Aqua Security, https://github.com/aquasecurity/trivy.

Docker Compose Documentation
Docker, "Docker Compose Documentation," Docker, https://docs.docker.com/compose/.

DevSecOps: A Quick Guide
P. Williams, "DevSecOps: Integrating Security into CI/CD Pipelines," TechBeacon, https://techbeacon.com/.

OWASP DevSecOps
OWASP, "DevSecOps: Integrating Security into DevOps," Open Web Application Security Project, https://owasp.org/www-project-devsecops/.

GitHub Documentation
GitHub, "GitHub Docs," GitHub, https://docs.github.com/en/github.