PROJECT: STREAMLINING SECURITY ACROSS ENVIRONMENTS WITH DEVSECOPS

PHASE 1 - PROBLEM ANALYSIS

College Name: Vemana Institute of Technology
Name: Monika P
CAN ID Number: CAN_33387138

ABSTRACT

Modern organizations face challenges ensuring security throughout software development and deployment processes. DevSecOps integrates security into DevOps workflows to automate, identify, and mitigate vulnerabilities early. This project aims to streamline security across environments by integrating security checks into the CI/CD pipeline. We will focus on automating vulnerability detection, secure code practices, and ensuring compliance. By leveraging DevSecOps principles and advanced tools, the project aims to enhance deployment security, minimize manual interventions, and meet organizational security and compliance requirements.

PROBLEM STATEMENT

Consider an organization deploying applications across various environments. The development, operations, and security teams face several critical challenges:

- **Manual Security Configuration:** Lack of automation leads to misconfigurations and vulnerabilities across deployments.
- **Vulnerability Management:** Difficulty identifying and fixing vulnerabilities across microservices.
- **Inefficient Security Practices:** Security assessments are often manual and late in the lifecycle, leading to delays.
- **Compliance Issues:** Inconsistent enforcement of security policies across environments complicates audit readiness.

KEY PARAMETERS IDENTIFIED

- **Deployment Issues:**

    - Lack of automated security scans.
    - Vulnerabilities due to improper image management.
- **CI/CD Bottlenecks:**
    - Manual security checks slow down releases.
- **User Needs:**
    - Automated and integrated security checks for deployments.
    - Scalable infrastructure with secure DevSecOps practices.

APPLICATION REQUIREMENTS

Application Structure:

pgsql

CopyEdit

project/

├── public/

│ ├── css/

│ ├── js/

│ └── index.html

├── server/

│ ├── controllers/

│ ├── models/

│ ├── routes/

│ └── server.js

├── package.json

└── README.md

Functional Requirements:

1. Integrate security scans into the CI/CD pipeline.
2. Automate vulnerability assessments and compliance checks.

3. Monitor and report on security status across environments.
4. Manage secure storage of deployment artifacts.

Non-Functional Requirements:

1. High availability and scalability for secure deployments.
2. Efficient integration with DevSecOps tools to minimize manual interventions.
3. Robust compliance with industry standards.

TOOLS IDENTIFIED

1. **Development:**
    - Docker: For containerized development.
2. **Version Control:**
    - Git/GitHub: For managing source code.
3. **CI/CD Pipeline:**
    - Jenkins/GitHub Actions: For automated build and deployment processes.
    - Kubernetes CLI (kubectl): For deployment orchestration.
4. **Security:**
    - OWASP ZAP, Snyk: For vulnerability scanning.
    - IBM Cloud Container Registry: For secure container storage.

FUTURE PLAN

1. **Automating CI/CD Security:**
    - Goal: Fully automate security scanning and remediation in the CI/CD process.
    - Tools: Jenkins, GitHub Actions, OWASP ZAP.
    - Plan: Automate scans, alert for vulnerabilities, and enforce secure code policies.
2. **Improving Cluster Security:**

- Goal: Harden Kubernetes clusters for enhanced security.
- Tools: Kubernetes, Istio, kubectl.
- Plan: Implement network policies, RBAC, and encryption.

3. **Strengthening Artifact Security:**
   - Goal: Ensure secure storage and management of deployment artifacts.
   - Tools: IBM Cloud Container Registry, OpenSSL.
   - Plan: Enable vulnerability scans and image signing.