DevOps Shack

Scenario-Based Real-Time Trivy Use Case

[Click Here To Enrol To Batch-5 | DevOps & Cloud DevOps]

Scenario: Continuous Vulnerability Management in a CI/CD Pipeline for a Web Application

Context

You are a DevOps engineer at a company that develops and maintains a web application deployed in Docker containers. The application is continuously updated with new features and bug fixes. Ensuring the security of the application is critical, especially as new vulnerabilities are discovered frequently. You need to implement a solution that continuously scans Docker images for vulnerabilities before they are deployed to production.

Objectives

1. Automate the security scanning of Docker images to detect vulnerabilities early in the CI/CD pipeline.
2. Ensure that no Docker image with critical vulnerabilities is deployed to production.
3. Provide developers with detailed reports on vulnerabilities so they can address issues promptly.

Solution

Implementing Trivy in the CI/CD Pipeline

Step-by-Step Implementation

1. **Set Up Trivy**

Ensure Trivy is installed on your CI/CD runners. If you are using Jenkins, GitHub Actions, or GitLab CI, you can use the following installation command within your pipeline script:

```
curl -sfL
https://raw.githubusercontent.com/aquasecurity/trivy/main/contrib/ins
tall.sh | sh
```

2. **Integrate Trivy in the Pipeline**

   Modify your CI/CD pipeline configuration to include a stage for Trivy scanning. Below is an example of a Jenkins pipeline script that integrates Trivy:

```
pipeline {
    agent any
    stages {
        stage('Checkout') {
            steps {
                git 'https://github.com/your-repo.git'
            }
        }
        stage('Build') {
            steps {
                sh 'docker build -t your-app:latest .'
            }
        }
        stage('Trivy Scan') {
            steps {
                script {
                    def scanResult = sh(script: 'trivy image --exit-
code 1 --severity HIGH,CRITICAL your-app:latest', returnStatus: true)
                    if (scanResult != 0) {
                        error("Vulnerabilities found in the Docker
image. Aborting the build.")
                    }
                }
            }
        }
        stage('Deploy') {
            when {
                branch 'main'
            }
            steps {
                sh 'docker push your-app:latest'
            }
        }
    }
    post {
        always {
            archiveArtifacts artifacts: '**/trivy-report.html',
allowEmptyArchive: true
        }
    }
}
```

3. **Reviewing and Addressing Vulnerabilities**

- **Report Generation**: Ensure Trivy generates a detailed HTML report that is archived by Jenkins for later review.
- **Developer Feedback**: If vulnerabilities are found, the pipeline will stop, and developers will be notified with details of the vulnerabilities. This can be done via email notifications, Slack messages, or directly in the CI/CD platform.

4. **Continuous Improvement**
   - **Regular Updates**: Regularly update Trivy's vulnerability database to ensure the latest vulnerabilities are checked.
   - **Custom Rules**: Customize the scanning process by creating `.trivyignore` files to ignore known non-critical vulnerabilities that you do not wish to block the pipeline.

Example Output

After running the pipeline, if Trivy detects vulnerabilities, the output might look like this:

```
2024-06-27T10:00:00.000Z	INFO	Need to update DB
2024-06-27T10:00:00.000Z	INFO	Downloading DB...
2024-06-27T10:00:10.000Z	INFO	Detected OS: alpine
2024-06-27T10:00:10.000Z	INFO	Detecting Alpine vulnerabilities...
2024-06-27T10:00:10.000Z	INFO	Number of language-specific files: 2
2024-06-27T10:00:10.000Z	INFO	Detecting gobinary vulnerabilities...

nginx:latest (alpine 3.13)
==========================================
Total: 5 (UNKNOWN: 0, LOW: 1, MEDIUM: 1, HIGH: 2, CRITICAL: 1)

+---------+------------------+------------------+-------------------+------
-----------+---------------------------------+
| LIBRARY | VULNERABILITY ID | SEVERITY         | INSTALLED VERSION | FIXED
VERSION   | TITLE                           |
+---------+------------------+------------------+-------------------+------
-----------+---------------------------------+
| apk-tools | CVE-2021-36159  | HIGH             | 2.12.1-r0         |
2.12.4-r0       | apk-tools: Out-of-bounds Read in |
|           |                  |                  |                   |
| apk_audit.c                       |
+---------+------------------+------------------+-------------------+------
-----------+---------------------------------+
| openssl  | CVE-2021-3712    | CRITICAL         | 1.1.1k-r0         |
1.1.1l-r0       | openssl: Read buffer overflow    |
|           |                  |                  |                   |
| in X509_issuer_and_serial_hash() |
+---------+------------------+------------------+-------------------+------
-----------+---------------------------------+
```

Benefits

1. **Early Detection**: By integrating Trivy into your CI/CD pipeline, vulnerabilities are detected early in the development cycle, reducing the risk of deploying insecure code.

2. **Automated Security**: Automating security scans ensures consistent and repeatable security checks without manual intervention.
3. **Developer Awareness**: Providing detailed reports helps developers understand and fix vulnerabilities promptly, fostering a security-first mindset.

Conclusion

Integrating Trivy into your CI/CD pipeline enhances the security of your web application by automating vulnerability scanning. This proactive approach helps maintain high security standards and ensures that only secure code is deployed to production.