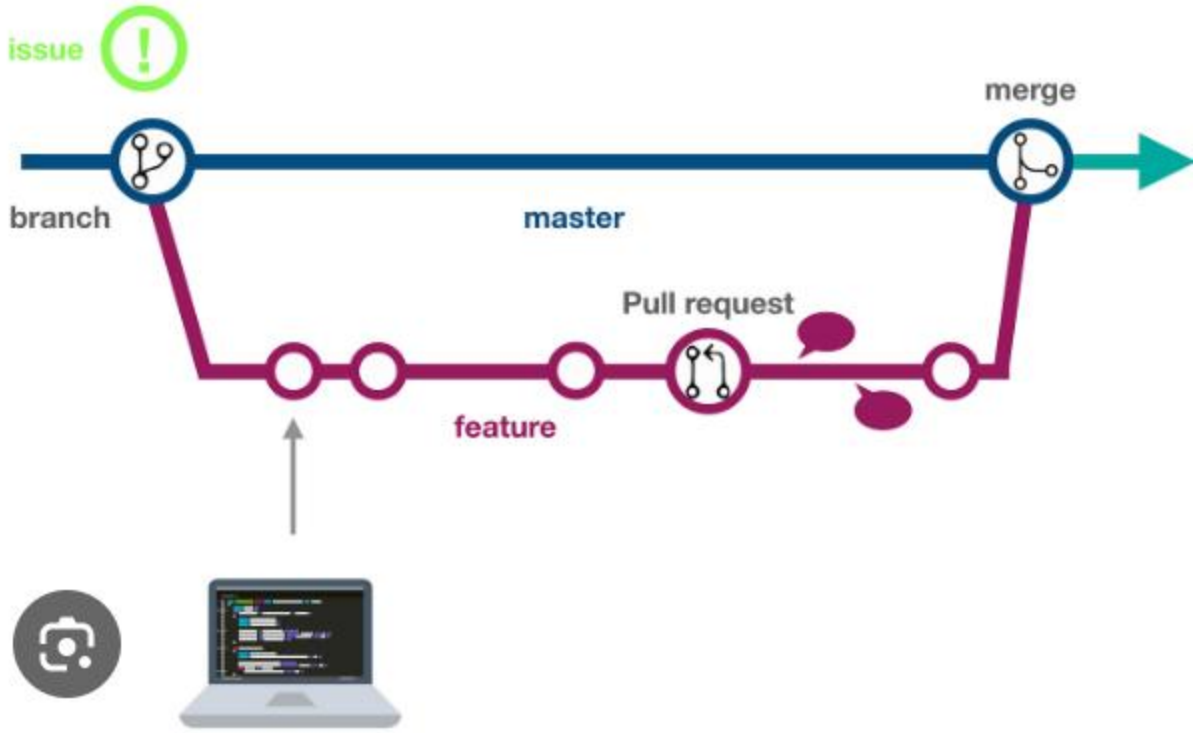
Project Overview

This project involves setting up a **Secure CI/CD pipeline** for a chosen open-source application. The pipeline integrates multiple security tools to ensure that best practices in security and software development are followed. The pipeline utilizes tools such as **GitHub Actions**, **SAST**, **DAST**, **SCA**, **Container Security**, and **IaC security scanninG , SIEM & Vulnerability Management** to perform static and dynamic analysis, enforce secure coding practices, and ensure the security of the deployed containers.

1. Repo : Owasp - Juice Shop : https://github.com/BilalBaree/Owasp_JuiceShop

2. Git Branching Strategy

- **Git Strategy Chosen**: **GitHub Flow**
    - **Justification**: GitHub Flow is simple, and its focus on pull requests aligns well with the goals of enforcing security checks and code review mechanisms. This strategy ensures that every change to the codebase goes through a review process before being merged, which allows for better control over the quality and security of the code.



3. CI Tool: GitHub Actions

- **CI Tool Chosen**: **GitHub Actions**
    - **Justification**: GitHub Actions provides seamless integration with GitHub repositories, allowing for automatic workflows for testing, building, and deploying code. It is highly customizable and allows the integration of various security tools into the CI/CD pipeline, which fits the project requirements.

–

4. Security Checks in CI

The following security checks are integrated into the CI pipeline to ensure that the code adheres to secure practices:

a. Code Reviews and Signed Commits

- **Enforce Code Reviews**: The pipeline ensures that code reviews are mandatory for pull requests before merging into the main branch.
- **Enforce Signed Commits**: Each commit in the pipeline must be signed to guarantee authenticity. This check ensures that only trusted contributors can make changes.

b. Role-Based Access Control (RBAC)

- **Access Control Enforcement**: The pipeline includes a check to verify that role-based access is configured correctly in the GitHub repository. It ensures that only authorized users have write access to critical branches.

c. Audit Logging

- **Pipeline Execution Logs**: Each pipeline run logs its execution details to an audit log, helping track security-related activities and actions.

```
#     auto-approve:
    # runs-on: ubuntu-latest
    # if: github.event_name == 'pull_request'
    # steps:
    #   - name: Check if PR author is trusted and target is main
    #     id: check
    #     run: |
    #       echo "PR author: ${{ github.event.pull_request.user.login }}"
    #       echo "Target branch: ${{ github.event.pull_request.base.ref }}"

    #       if [[ "${{ github.event.pull_request.user.login }}" ==
"trusted-user" && "${{ github.event.pull_request.base.ref }}" == "main" ]];
then
    #         echo "approve=true" >> $GITHUB_OUTPUT
    #       else
    #         echo "approve=false" >> $GITHUB_OUTPUT
    #       fi
```

```
      - name: Enforce code reviews (Only on pull request)
        if: github.event_name == 'pull_request'
        uses: hmarr/auto-approve-action@v2
        with:
          github-token: ${{ secrets.GITHUB_TOKEN }}

      - name: Enforce signed commits
        run: |
          git log ${{ github.event.before }}..${{ github.sha }} --
pretty=format:'%h %G? %an' | while read line; do
            commit_hash=$(echo $line | cut -d ' ' -f 1)
            commit_status=$(echo $line | cut -d ' ' -f 2)
            if [[ "$commit_status" != "G" ]]; then
              echo "❌ Commit $commit_hash is not signed"
              exit 1
            fi
          done

  # 2. Role-based Access Control
  access_control:
    runs-on: ubuntu-latest
    steps:
      - name: Check role-based access control
        run: |
          if [[ "$(curl -H "Authorization: Bearer
${{ secrets.GITHUB_TOKEN }}" \
          -s
"https://api.github.com/repos/${{ github.repository }}/collaborators" | jq
'.[] | select(.permissions.push == true)' | wc -l)" -lt 1 ]]; then
            echo "❌ Role-based access is not configured properly."
            exit 1
          Fi
 audit_log:
    runs-on: ubuntu-latest
    steps:
      - name: Log pipeline run
        run: |
          echo "$(date) - CI/CD pipeline executed for repo:
${{ github.repository }}" >> ./audit_log.txt
          cat ./audit_log.txt

      - name: Upload Audit Log
```

```
        uses: actions/upload-artifact@v4
        with:
          name: audit-log
          path: audit_log.txt
```

5. SAST (Static Application Security Testing)

The pipeline integrates two key **SAST** tools to identify vulnerabilities in the codebase:

a. TruffleHog (Secrets Scanning)

- **Purpose**: Detect hardcoded secrets, such as API keys or passwords.
- **Positive Case**: No secrets are found in the codebase, and the pipeline proceeds without issues.
- **Negative Case**: Secrets are found, and the pipeline fails, preventing potentially sensitive data from being exposed.

b. Bandit (SQL Injection Detection)

- **Purpose**: Detect potential SQL injection vulnerabilities within the code.
- **Positive Case**: No issues are found, and the pipeline proceeds.
- **Negative Case**: SQL injection vulnerabilities are detected, and the pipeline halts, requiring a fix before proceeding.

```
sast_analysis:
   runs-on: ubuntu-latest
   steps:
     - name: Checkout code
       uses: actions/checkout@v3

     - name: Set up Python
       uses: actions/setup-python@v5
       with:
         python-version: '3.x'

     - name: Install tools
       run: pip install bandit trufflehog jq

     - name: Secrets Scanning (TruffleHog)
       run: |
         trufflehog --json . > secrets_scan_results.json || true
         if [[ $(jq length secrets_scan_results.json) -gt 0 ]]; then
```

```
        echo "❌ Secrets found!"
        cat secrets_scan_results.json
        exit 1
      else
        echo "☑ No secrets found."
      fi

  - name: Upload TruffleHog Report
    uses: actions/upload-artifact@v4
    with:
      name: secrets-scan
      path: secrets_scan_results.json

  - name: SQL Injection Detection (Bandit)
    run: |
      bandit -r . -f json -o bandit_report.json -t B107
      if grep -q "issue" bandit_report.json; then
        echo "❌ SQL Injection issues found!"
        cat bandit_report.json
        exit 1
      fi

  - name: Upload Bandit Report
    uses: actions/upload-artifact@v4
    with:
      name: bandit-report
      path: bandit_report.json
```

6. DAST (Dynamic Application Security Testing)

Although DAST tools are not fully implemented, **SQLMap** and **OWASP ZAP** are integrated into the pipeline to check for runtime vulnerabilities:

a. SQLMap (SQL Injection Detection)

- **Purpose**: Identify SQL injection vulnerabilities in the deployed application.
- **Positive Case**: No SQL injection issues are found, and the pipeline proceeds.
- **Negative Case**: SQL injection vulnerabilities are identified, and the pipeline fails, preventing deployment.

b. OWASP ZAP (Cross-Site Scripting Detection)

- **Purpose**: Scan for cross-site scripting (XSS) vulnerabilities in web applications.
- **Positive Case**: No XSS vulnerabilities are detected.
- **Negative Case**: XSS vulnerabilities are found, and the pipeline fails until the issues are resolved.

```
dast_scan:
   runs-on: ubuntu-latest
   steps:
     - name: Checkout code
       uses: actions/checkout@v3

     - name: Set up Docker Buildx
       uses: docker/setup-buildx-action@v2

     - name: Pull and start Juice Shop container
       run: |
         docker pull bkimminich/juice-shop
         docker run --rm -d -p 3000:3000 bkimminich/juice-shop
         # Wait for the application to be ready
         for i in {1..60}; do
           if curl --silent --fail http://localhost:3000; then
             echo "Juice Shop is ready."
             break
```

```
            else
              echo "Waiting for Juice Shop to start..."
              sleep 5
            fi
          done

      - name: Install tools
        run: |
          pip install sqlmap
          sudo apt-get update
          sudo apt-get install -y openjdk-11-jre python3
          wget
https://github.com/zaproxy/zaproxy/releases/download/v2.16.1/ZAP_2.16.1_Lin
ux.tar.gz
          tar -xzf ZAP_2.16.1_Linux.tar.gz
          sudo mv ZAP_2.16.1 /opt/zaproxy
          sudo ln -s /opt/zaproxy/zap.sh /usr/local/bin/zap

      - name: Run SQLMap for SQLi
        run: sqlmap -u "http://localhost:3000" --batch --risk=3 --level=5

      - name: Run OWASP ZAP for XSS
        run: zap -cmd -quickurl http://localhost:3000 -quickout
zap_report.html

      - name: Upload ZAP Report
        uses: actions/upload-artifact@v4
        with:
          name: zap-report
          path: zap_report.html
```

7. SCA (Software Composition Analysis)

The pipeline uses **Snyk** to scan for vulnerable third-party dependencies:

a. Snyk (Vulnerable Packages Scan)

- **Purpose**: Scan the code for insecure third-party dependencies.
- **Positive Case**: No critical vulnerabilities are found, allowing the pipeline to proceed.
- **Negative Case**: Critical vulnerabilities are found, and the pipeline halts to fix the issues.

b. Signed Libraries Verification

- **Purpose**: Ensure that libraries used in the project are signed to prevent tampering.
- **Positive Case**: All libraries are signed, and the pipeline proceeds.
- **Negative Case**: Unsigned libraries are detected, and the pipeline fails.

```
# 5. SCA Tool: Snyk
  sca_scan:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18.19.0'

      - name: Install dependencies
        run: npm install

      - name: Scan for vulnerable packages (Snyk)
        uses: snyk/actions/node@master
        with:
          args: test . --severity-threshold=high
        env:
          SNYK_TOKEN: ${{ secrets.SNYK_TOKEN }}
```

```
      - name: Use signed libraries verification
        run: bash scripts/verify-signatures.sh
        continue-on-error: true
```

8. Container Security

The pipeline integrates security scanning for containers, specifically focusing on Docker images:

a. Clair (Root Access Check)

- **Purpose**: Scan the Docker image for root access privileges.
- **Positive Case**: No root access is found, and the pipeline continues.
- **Negative Case**: Root access is detected, and the pipeline fails.

b. Trivy (Read-Only File System)

- **Purpose**: Ensure that containers use read-only file systems for security.
- **Positive Case**: The container adheres to the read-only file system policy.
- **Negative Case**: Violations are found, and the pipeline fails until addressed.

```
 # 7. Container Security Scanning
  container-security-scan:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v2

      - name: Build Docker image
        run: |
          docker build -t ${{ secrets.DOCKERHUB_USERNAME }}/juice-
shop:latest .

      - name: Scan Docker image for Root Access (Clair)
        run: |
          curl -sL
https://github.com/coreos/clair/releases/download/v2.0.9/clairctl-linux-
amd64 -o clairctl
          chmod +x clairctl
```

```
          sudo mv clairctl /usr/local/bin/
          clairctl analyze --local --image
${{ secrets.DOCKERHUB_USERNAME }}/juice-shop:latest
          clairctl report ${{ secrets.DOCKERHUB_USERNAME }}/juice-
shop:latest
        continue-on-error: true

      - name: Scan image for Read-Only File System violations (Trivy)
        run: |
          docker run --rm -v /var/run/docker.sock:/var/run/docker.sock -v
$(pwd):/workspace aquasecurity/trivy fs --severity CRITICAL,HIGH --exit-
code 1 --no-progress --ignore-unfixed .
        continue-on-error: true

      - name: Push Docker image to DockerHub
        if: success()
        run: |
          docker login -u ${{ secrets.DOCKERHUB_USERNAME }} -p
${{ secrets.DOCKERHUB_TOKEN }}
          docker push ${{ secrets.DOCKERHUB_USERNAME }}/juice-shop:latest

  # 8. Generate and Upload Reports
  upload-reports:
    runs-on: ubuntu-latest
    steps:
      - name: Upload Clair report
        uses: actions/upload-artifact@v4
        with:
          name: clair-report
          path: clair-report.html

      - name: Upload Trivy report
        uses: actions/upload-artifact@v4
        with:
          name: trivy-report
          path: trivy-report.json
```

9. IaC (Infrastructure as Code) Security

The pipeline integrates **Checkov** and **Conftest** to enforce best practices and security for infrastructure as code:

a. Checkov (Misconfigurations in IaC)

- **Purpose**: Scan Terraform configurations for misconfigurations.
- **Positive Case**: No misconfigurations are found.
- **Negative Case**: Misconfigurations are found, and the pipeline fails.

b. Conftest (Compliance and Least Privilege)

- **Purpose**: Ensure compliance with policies such as least privilege.
- **Positive Case**: Policy compliance is confirmed, and the pipeline continues.
- **Negative Case**: Policy violations are found, and the pipeline halts for remediation.

```
# 9. IaC Security Scan
 iac-security-scan:
   runs-on: ubuntu-latest
   steps:
     - name: Checkout code
       uses: actions/checkout@v3

     - name: Set up Python
       uses: actions/setup-python@v4
       with:
         python-version: '3.x'

     - name: Install Terraform
       uses: hashicorp/setup-terraform@v2
       with:
         terraform_version: 1.4.6

     - name: Install Checkov
       run: pip install checkov
```

```
      - name: Install Conftest
        run: |
          CONFTAG=$(curl -s https://api.github.com/repos/open-policy-
agent/conftest/releases/latest | grep tag_name | cut -d '"' -f 4)
          wget https://github.com/open-policy-
agent/conftest/releases/download/${CONFTAG}/conftest_${CONFTAG#v}_Linux_x86
_64.tar.gz
          tar -xzf conftest_${CONFTAG#v}_Linux_x86_64.tar.gz
          sudo mv conftest /usr/local/bin

      - name: Run Checkov for misconfigurations
        run: |
          checkov -d ./terraform --skip-check CKV_AWS_355 --skip-check
CKV_SECRET_1

      - name: Run Conftest with least privilege policy
        run: |
          conftest test terraform/main.tf --policy policies

      - name: Check for secrets in Terraform using Checkov
        run: |
          checkov -d ./terraform --check CKV_SECRET_1
```

Integration Of DataDog (SIEM TOOL)

```
- name: Capture test results and send to Datadog
  env:
    DD_API_KEY: ${{ secrets.DATADOG_API_KEY }}
  run: |
    OUTPUT=$(npm test || true)
    curl -X POST "https://http-intake.logs.datadoghq.eu/v1/input" \
      -H "Content-Type: application/json" \
      -H "DD-API-KEY: $DD_API_KEY" \
      -d "{\"message\": \"$OUTPUT\", \"ddsource\": \"github-actions\",
\"service\": \"ci-tests\"}"
```

Integration Of Vulnerability Management Tools (CodeQL & Dependabot)

CodeQL

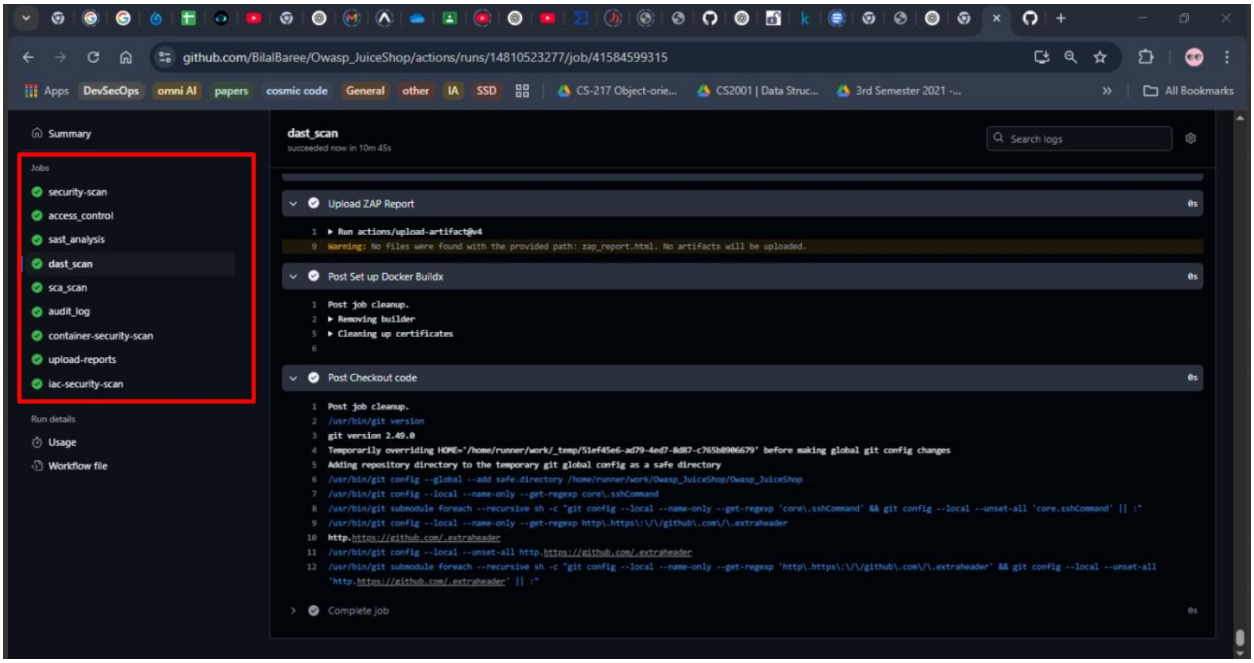
```yaml
name: "CodeQL Scan"

on:
  push:
  pull_request:

jobs:
  analyze:
    name: Analyze
    runs-on: ubuntu-latest
    permissions:
      actions: read
      contents: read
      security-events: write
    strategy:
      fail-fast: false
      matrix:
        language: [ 'javascript-typescript' ]
    steps:
    - name: Checkout repository
      uses: actions/checkout@11bd71901bbe5b1630ceea73d27597364c9af683
#v4.2.2
    - name: Initialize CodeQL
      uses: github/codeql-action/init@v3
      with:
        languages: ${{ matrix.language }}
        queries: security-extended
        config: |
          paths-ignore:
            - 'data/static/codefixes'
    - name: Autobuild
      uses: github/codeql-action/autobuild@v3
    - name: Perform CodeQL Analysis
      uses: github/codeql-action/analyze@v3
```

Dependabot

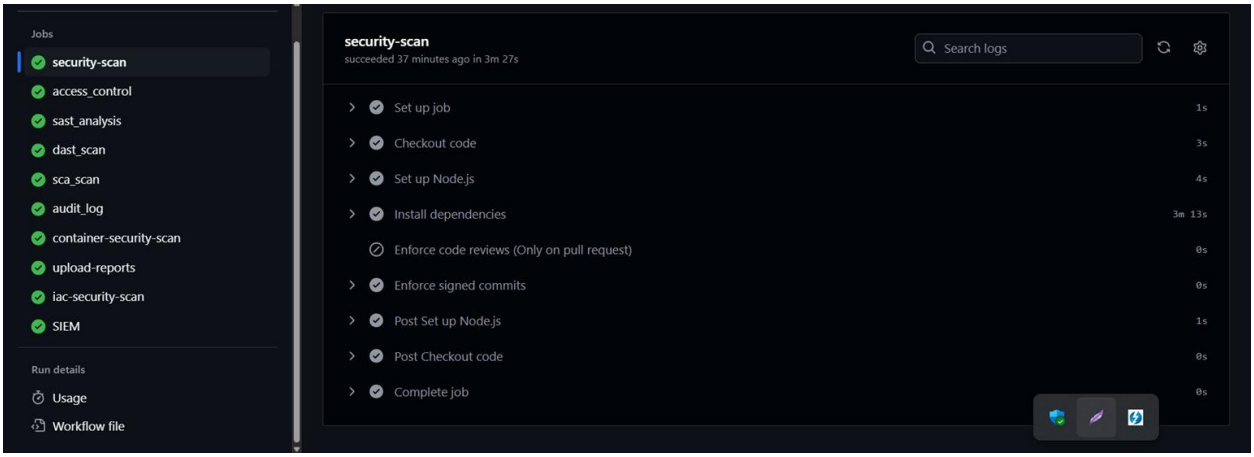
```
version: 1
update_configs:
  - package_manager: "javascript"
    directory: "/"
    update_schedule: "live"
    target_branch: "develop"
    default_reviewers:
      - "bkimminich"
    default_labels:
      - "dependencies"
    ignored_updates:
      - match:
          dependency_name: "express-jwt"
          version_requirement: "0.1.3"
      - match:
          dependency_name: "sanitize-html"
          version_requirement: "1.4.2"
      - match:
          dependency_name: "unzipper"
          version_requirement: "0.9.15"
      - match:
          dependency_name: "jsonwebtoken"
          version_requirement: "0.4.0"
  - package_manager: "javascript"
    directory: "/frontend"
    update_schedule: "live"
    target_branch: "develop"
    default_reviewers:
      - "bkimminich"
    default_labels:
      - "dependencies"
```

ScreenShots

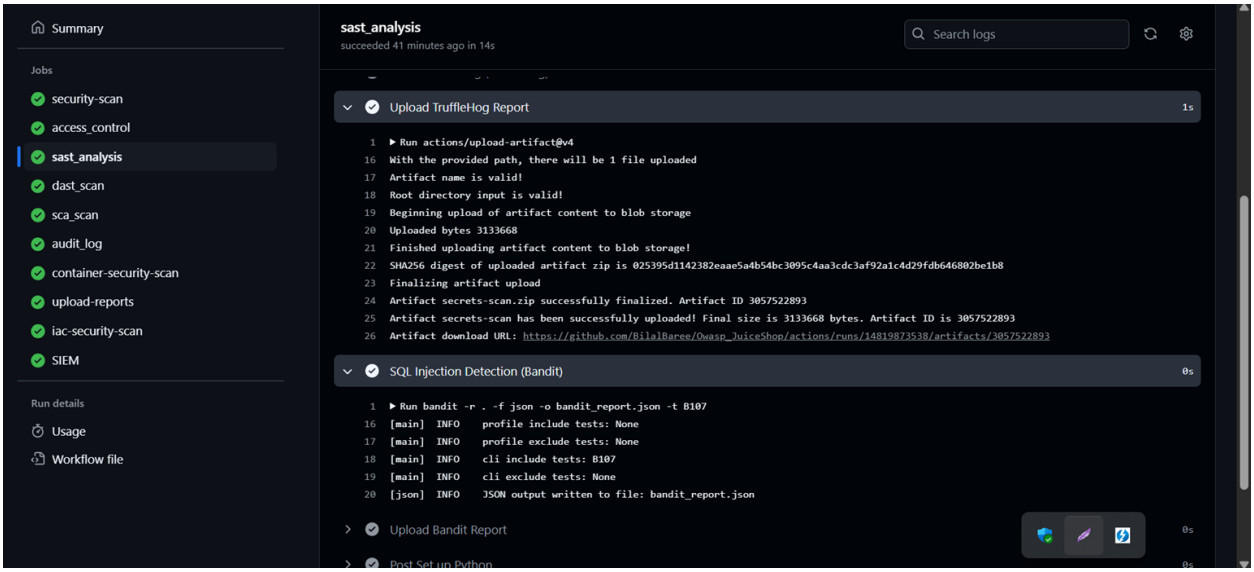
Pipeline Execution:



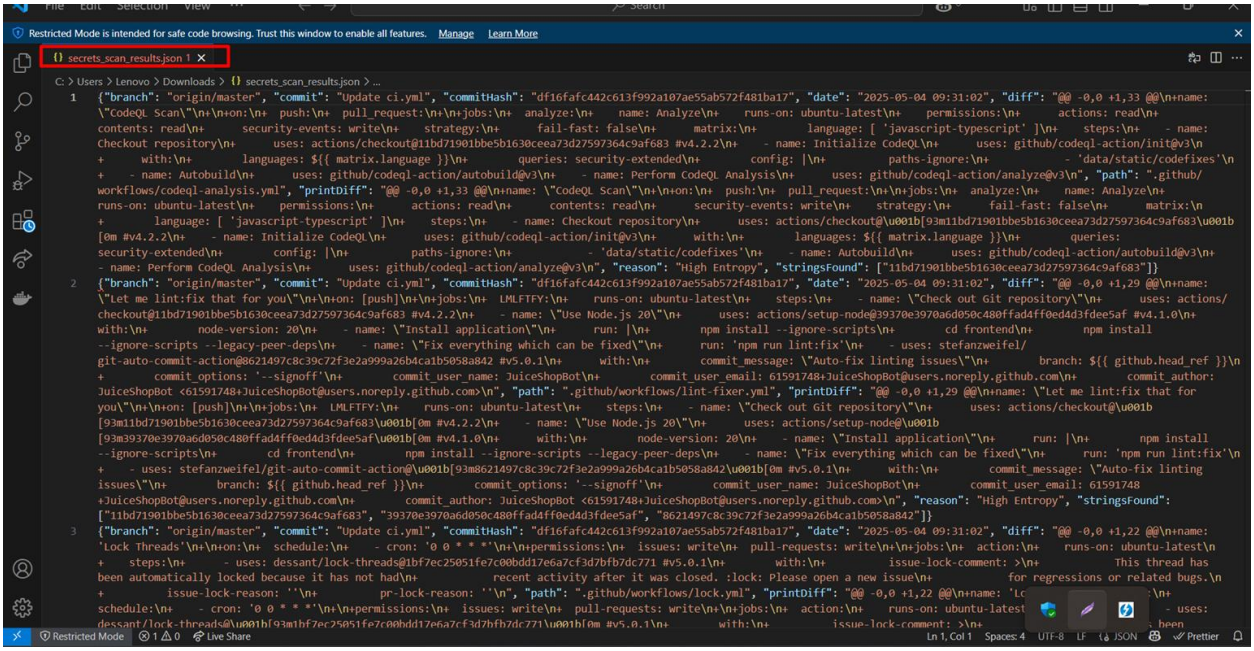
Security Check:



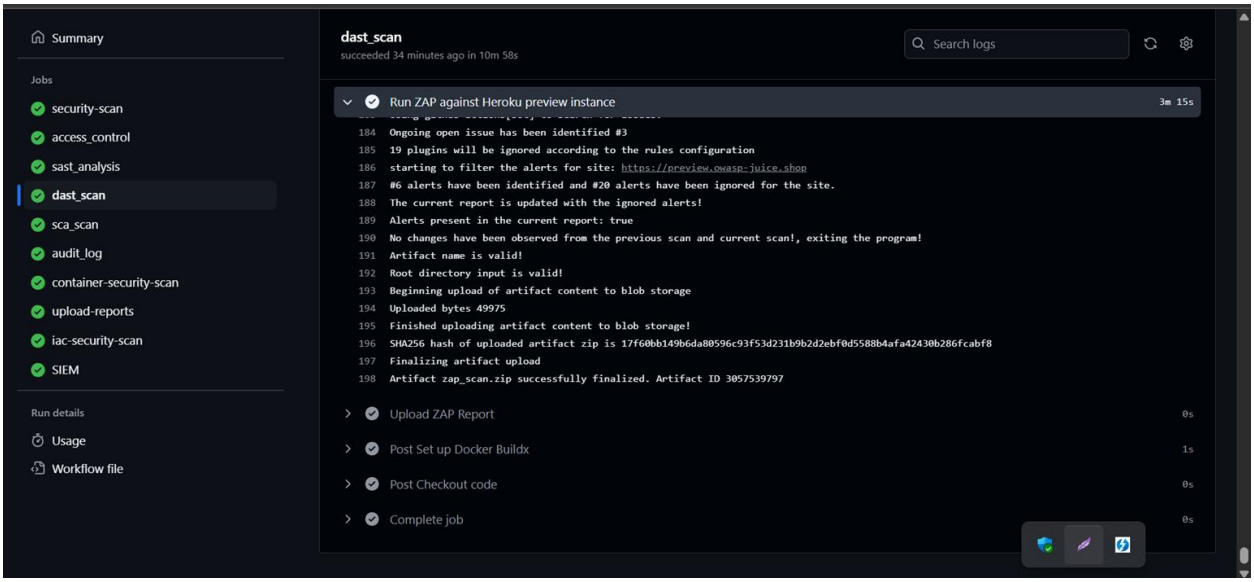
SAST Analysis (Bandit & Trufflehog):



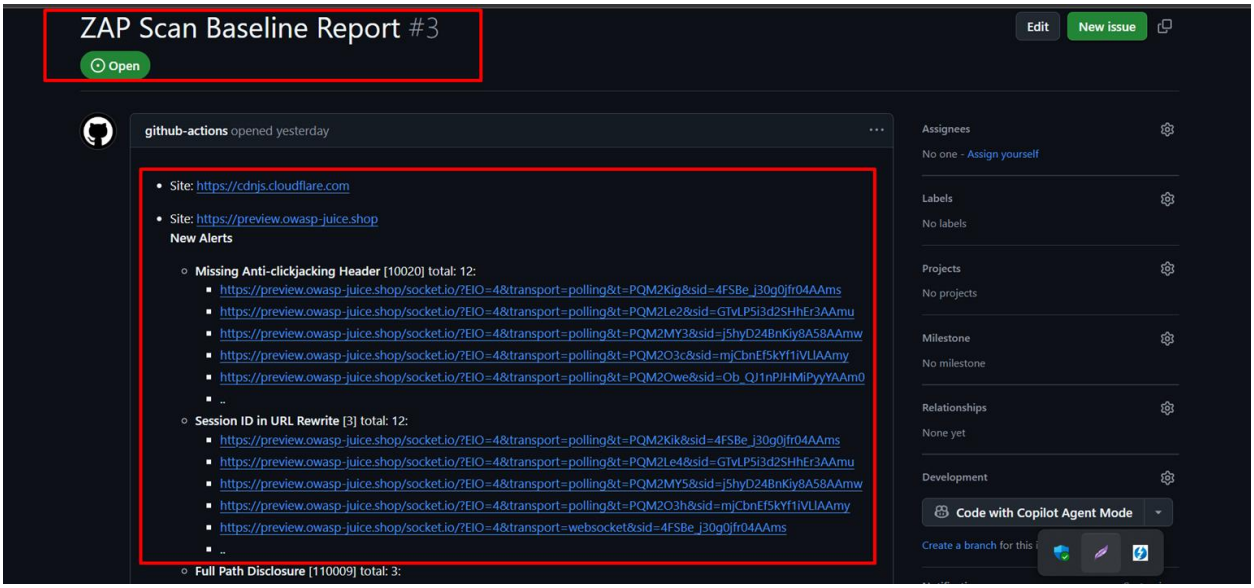
Bandit & Trufflehog SAST Analysis Reports:



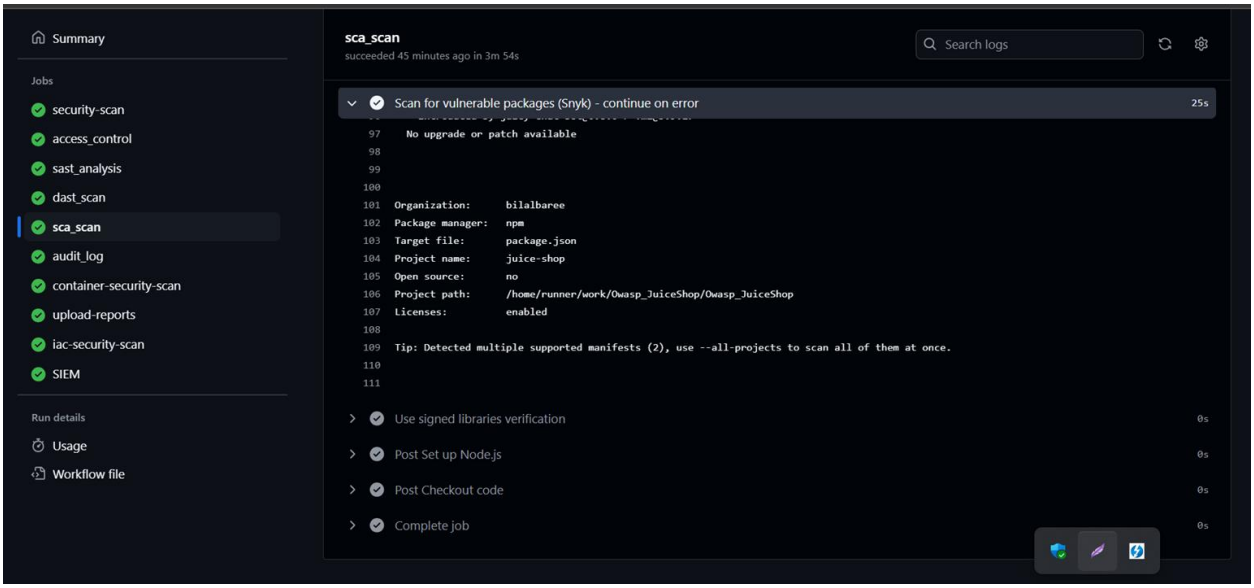
DAST Analysis(SQLMap & Owasp ZAP):



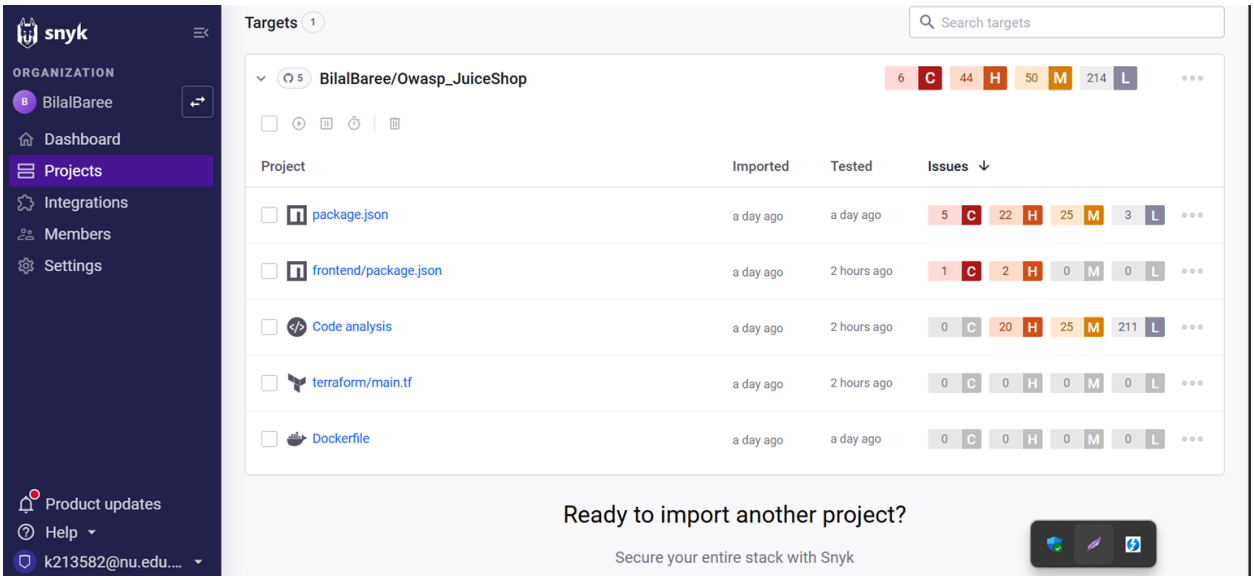
DAST Analysis Reports:



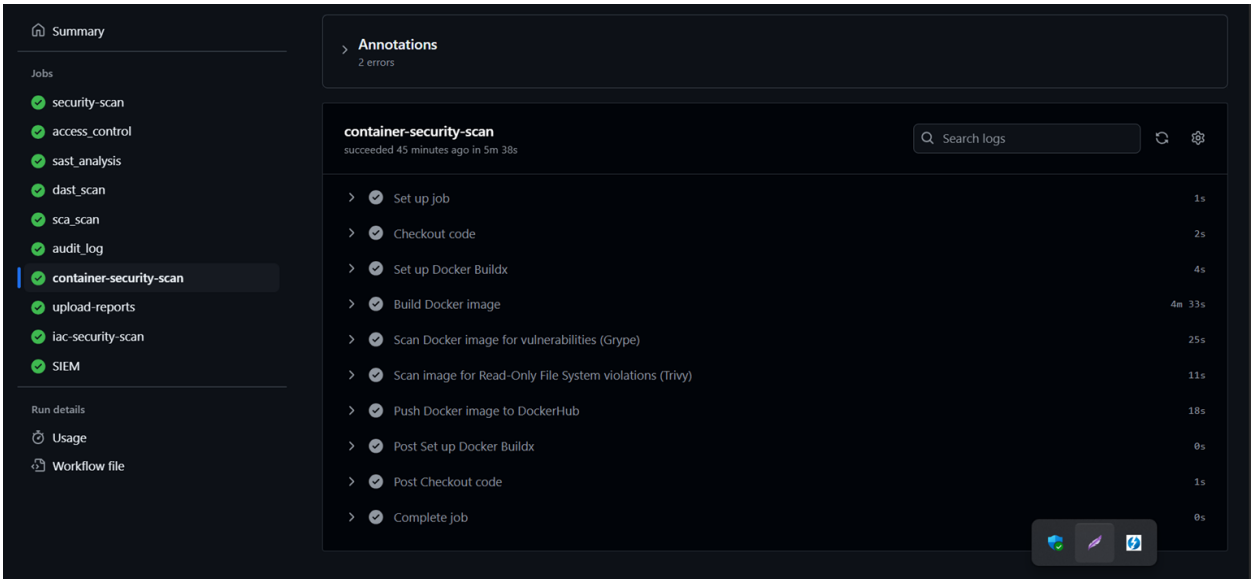
SCA (Snyk):



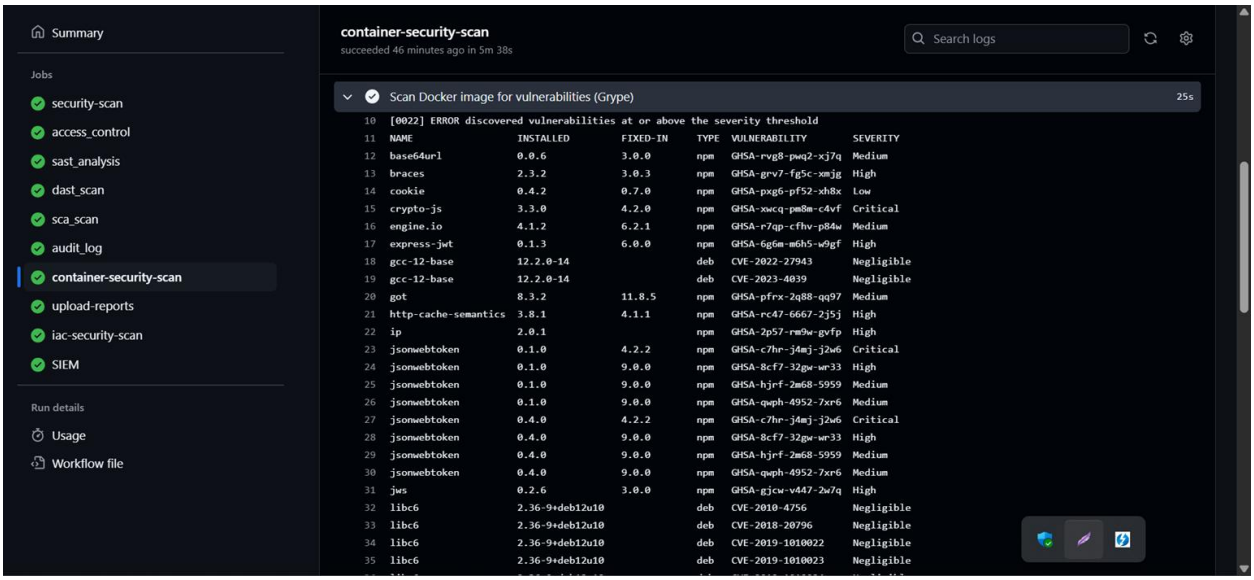
SCA Analysis Reports:

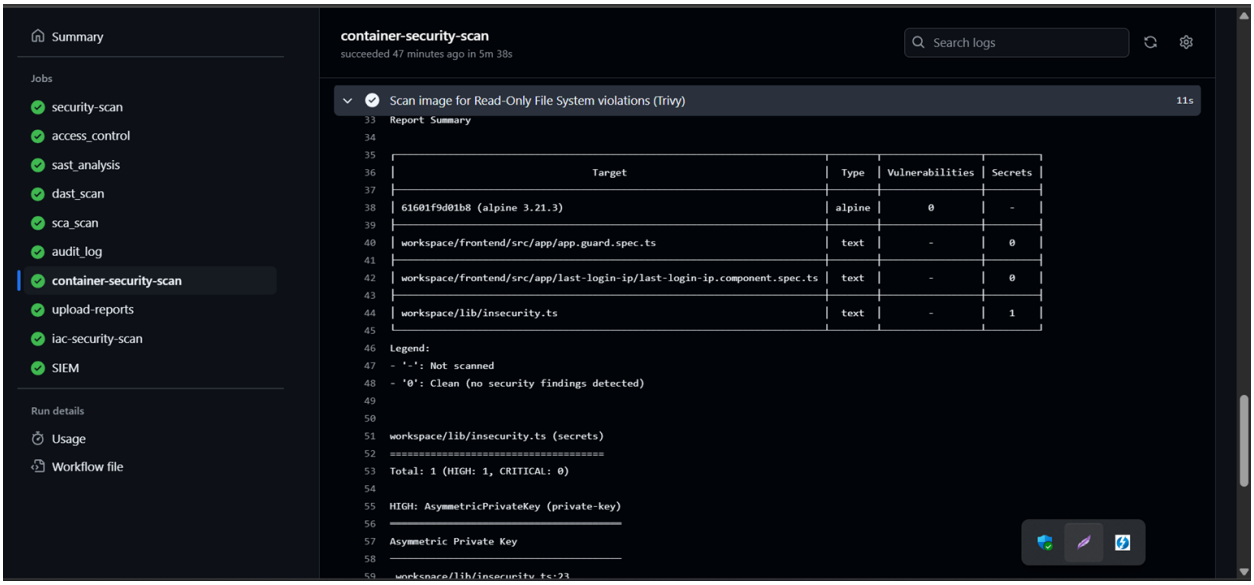


Container & Security (Grype & Trivy):

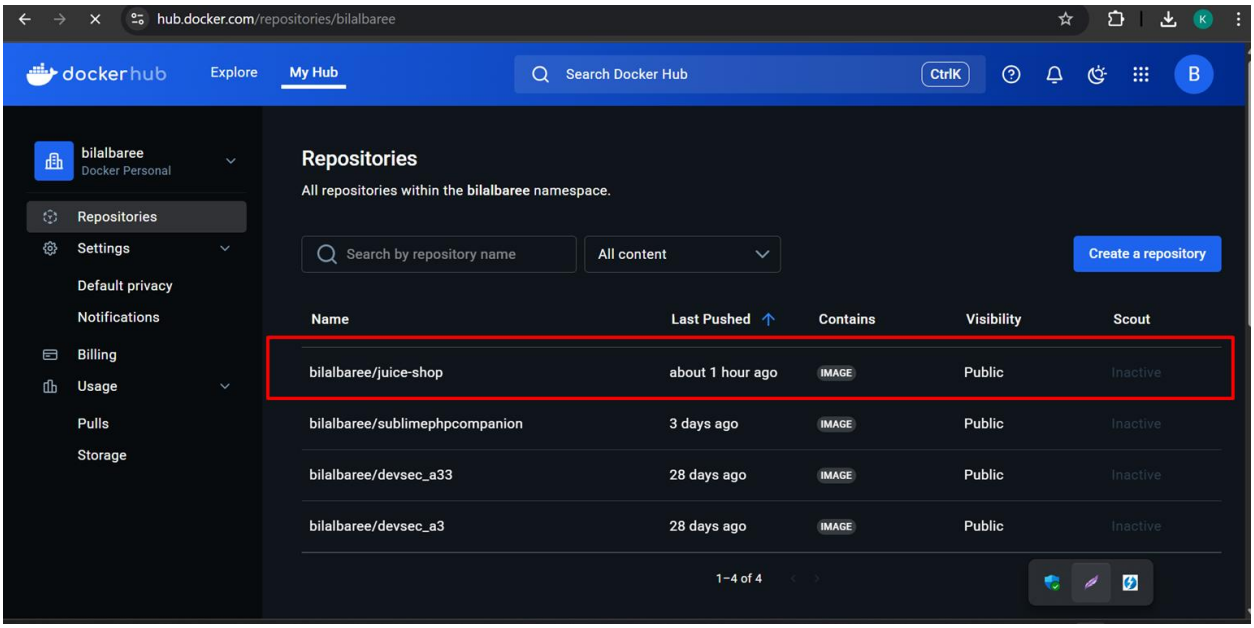


Grype & Trivvy Reports :



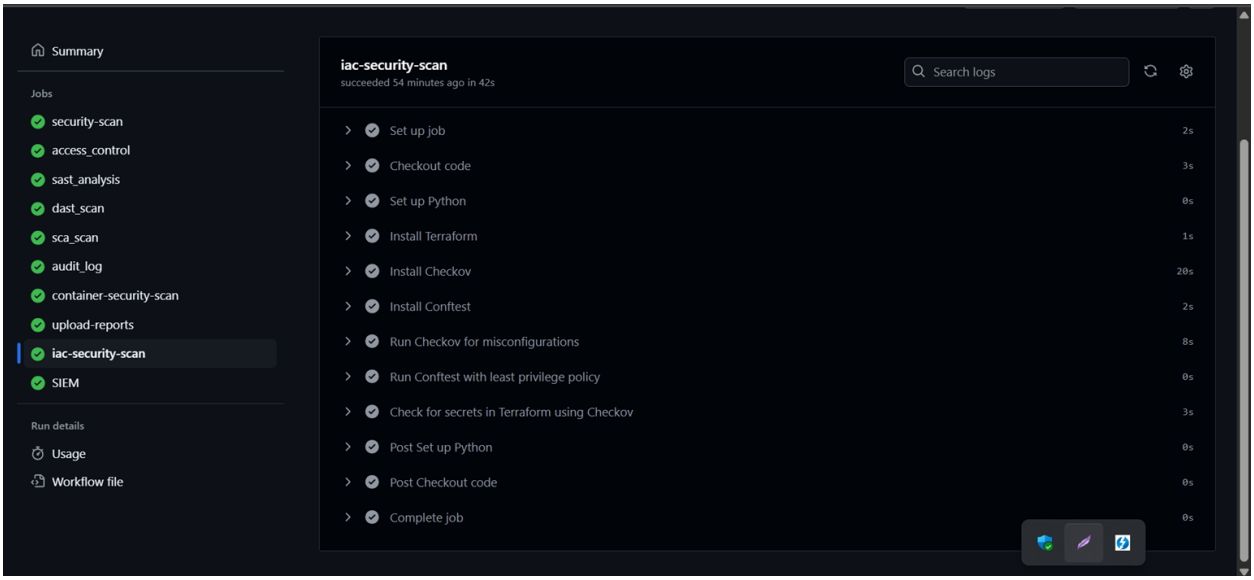


DockerHub Image Upload:

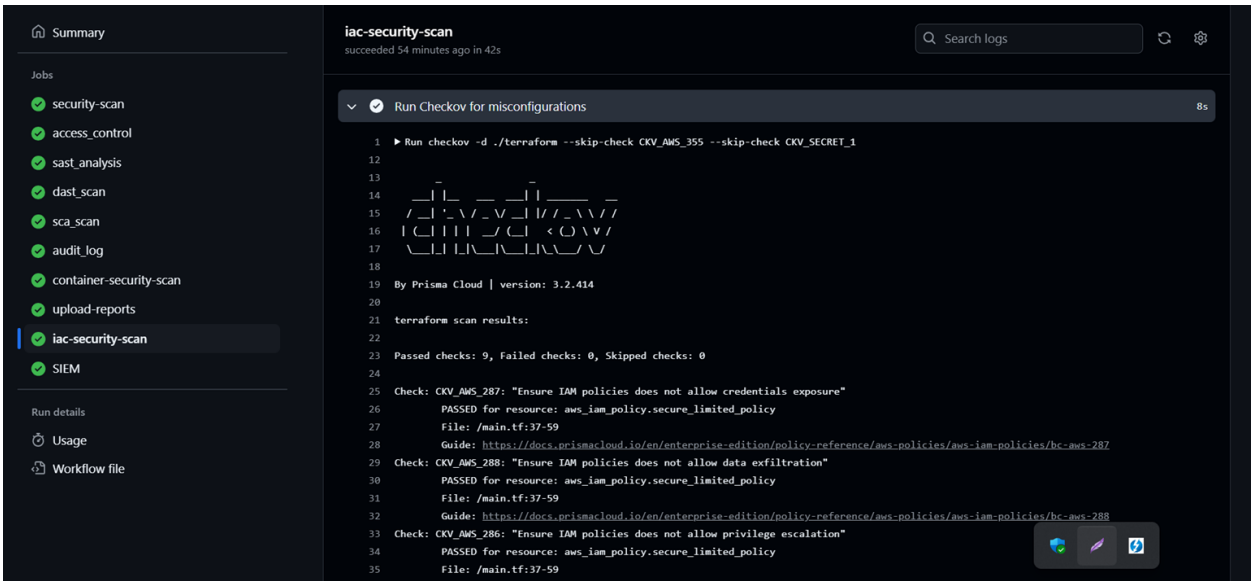


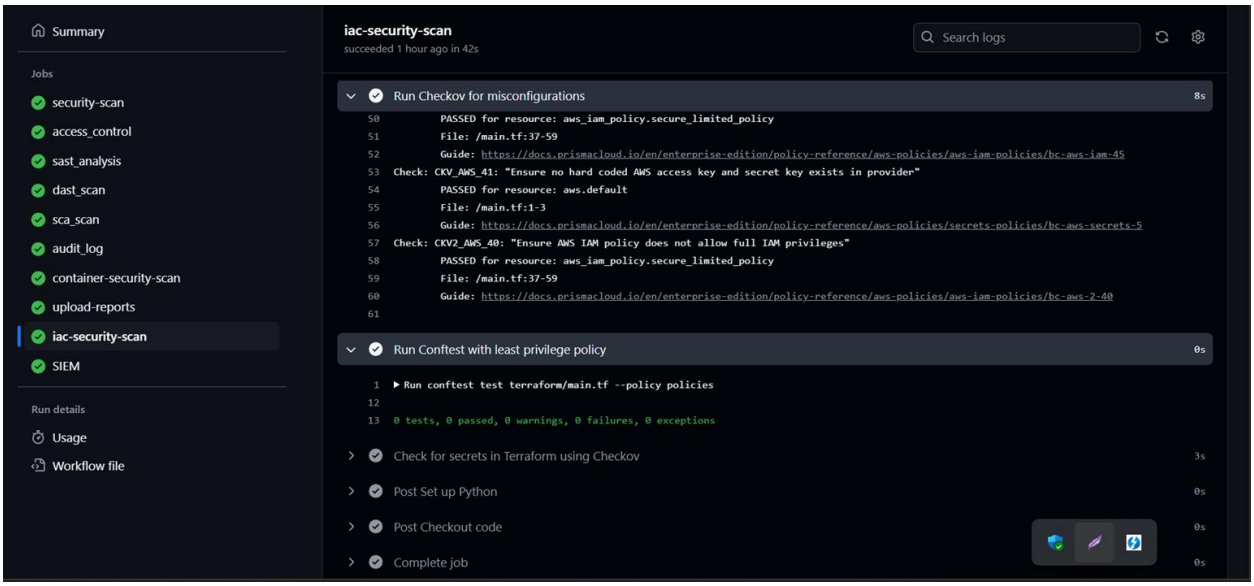
IaC Security (Terraform):

I cannot run EC2 instances (not free just uploading workflow file)

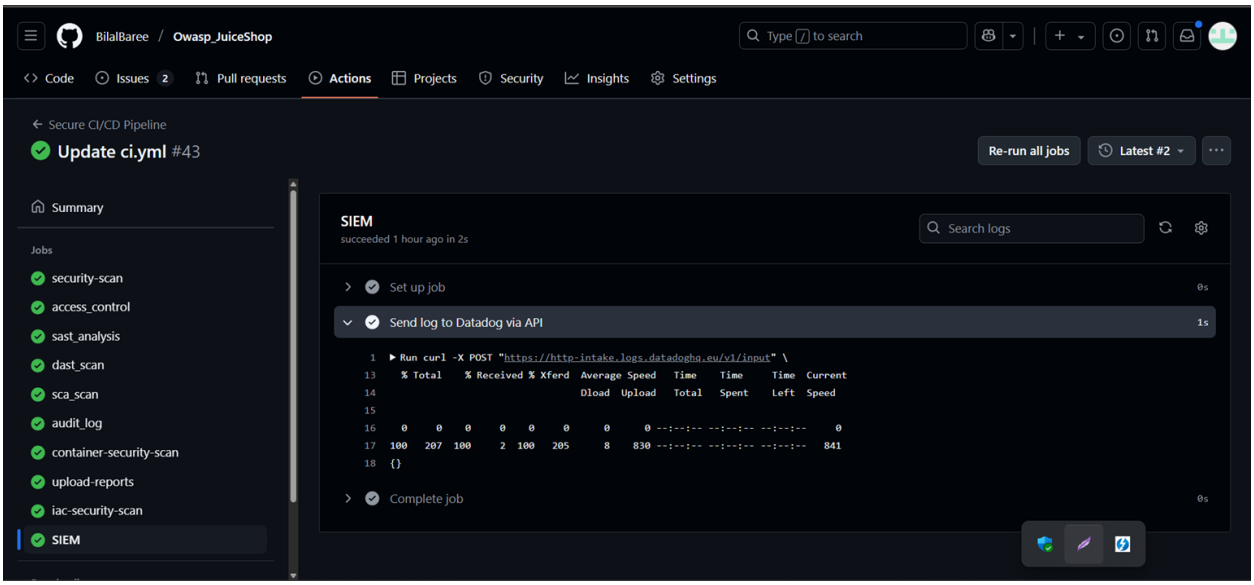


Conftest and Checkov for OPA and Policy Review/Enforcement:

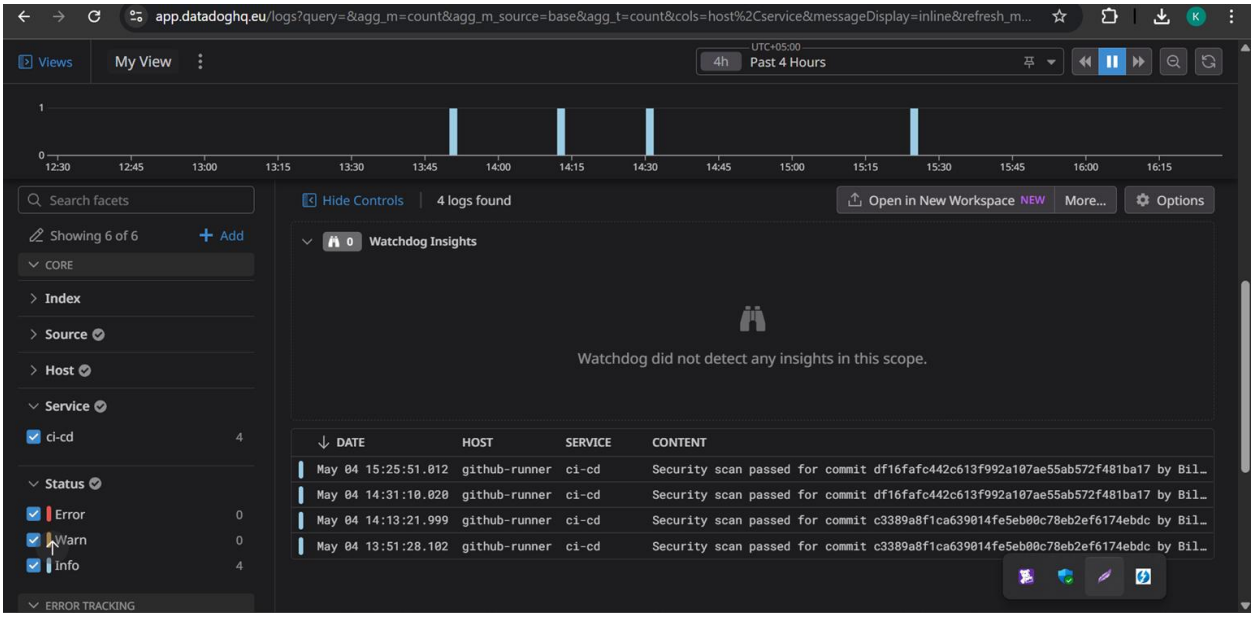




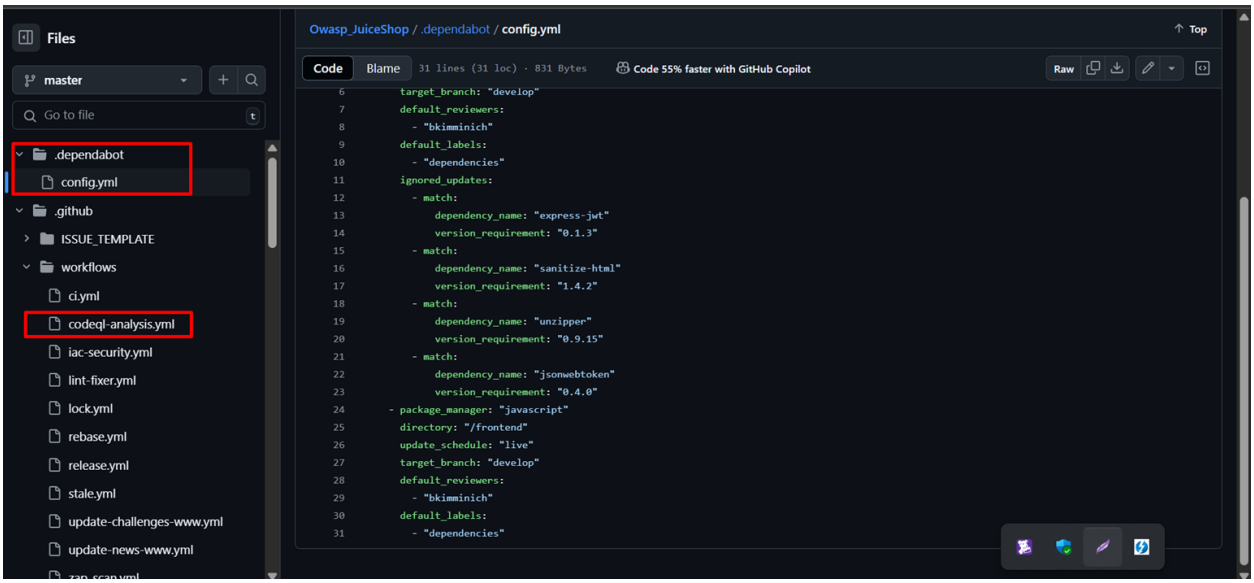
SIEM Tool Integration (DataDog):



DataDog Log Explorer :



Vulnerability Management (CodeQL & Dependabot):


Files
master
Go to file
.dependabot
config.yml
.github
ISSUE_TEMPLATE
workflows
ci.yml
codeql-analysis.yml
iac-security.yml
lint-fixer.yml
lock.yml
rebase.yml
release.yml
stale.yml
update-challenges-www.yml
update-news-www.yml
Owasp_JuiceShop / .dependabot / config.yml
Top
Code
Blame
31 lines (31 loc) · 831 Bytes
Code 55% faster with GitHub Copilot
Raw
target_branch: "develop"
default_reviewers:
- "bkimminich"
default_labels:
- "dependencies"
ignored_updates:
- match:
dependency_name: "express-jwt"
version_requirement: "0.1.3"
- match:
dependency_name: "sanitize-html"
version_requirement: "1.4.2"
- match:
dependency_name: "unzipper"
version_requirement: "0.9.15"
- match:
dependency_name: "jsonwebtoken"
version_requirement: "0.4.0"
- package_manager: "javascript"
directory: "/frontend"
update_schedule: "live"
target_branch: "develop"
default_reviewers:
- "bkimminich"
default_labels:
- "dependencies"