Nama : Muhammad Faturrohman Sugiyarto

NPM : 2019101631

Kelas : IV Rekayasa Perangkat Lunak Kriptografi

Link : https://github.com/fixxall/BadCRUD\_UAS

JAWABAN UJIAN AKHIR SEMESTER

PENGUJIAN PERANGKAT LUNAK

1. Test Case Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **TEST CASE** | **INPUT VALUE** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **PASS/FAIL** |
| T1 | Valid Credentials | Username: admin  Password: nimda666! | Successful Login | Successful Login | PASS |
| T2 | Invalid Username | Username: user  Password: nimda666! | Error Message | Message output “Wrong usename or password” | PASS |
| T3 | Invalid Password | Username: admin  Password: invalid | Error Message | Message output “Wrong usename or password” | PASS |
| T4 | Incorrect Credentials | Username: user  Password: user123 | Error Message | Message output “Wrong usename or password” | PASS |
| T5 | Password Case-Insensitive | Username: admin  Password: Nimda666! | Error Message | Message output “Wrong usename or password” | PASS |
| T6 | Account Lockout | Multiple failed login attempts | Account Lockout Message | Doesn’t have any respon. Just get error message same. Message output “Wrong usename or password” | FAIL |
| T7 | SQL Injection Security Testing | Username:” “’|#--  Password: ” “’|#-- | Error Message | Fatal error: Uncaught PDOException: SQLSTATE | FAIL |

1. Langkah otomasi:
2. T1 - Valid Credentials

Steps:

- Navigate to url http://localhost/login.php

- Verify sign in page is visible

- Enter valid username and password

- Click “Sign in” button

- Verify that logged in into dashboard as username

Source:

/testing/LoginTestCase/T1\_test.py

1. T2 - Invalid Username

Steps:

- Navigate to url http://localhost/login.php

- Verify sign in page is visible

- Enter invalid username and valid password

- Click “Sign in” button

- Verify that message “Wrong usename or password” appear on website

Source:

/testing/LoginTestCase/T2\_test.py

1. T3 - Invalid Password

Steps:

- Navigate to url http://localhost/login.php

- Verify sign in page is visible

- Enter valid username and invalid password

- Click “Sign in” button

- Verify that logged in into dashboard as username

- Click “Sign out” button

- Verify that message “Wrong usename or password” appear on website

Source:

/testing/LoginTestCase/T3\_test.py

1. T4 - Incorrect Credentials

Steps:

- Navigate to url http://localhost/login.php

- Verify sign in page is visible

- Enter invalid username and invalid password

- Click “Sign in” button

- Verify that logged in into dashboard as username

- Click “Sign out” button

- Verify that message “Wrong usename or password” appear on website

Source:

/testing/LoginTestCase/T4\_test.py

1. T5 - Password Case-Insensitive

Steps:

- Navigate to url http://localhost/login.php

- Verify sign in page is visible

- Enter valid username and valid password with Case Insensitive like Capitalized on password or username

- Click “Sign in” button

- Verify that message “Wrong usename or password” appear on website

Source:

/testing/LoginTestCase/T5\_test.py

1. T6 - Account Lockout

Steps:

- Navigate to url http://localhost/login.php

- Verify sign in page is visible

- Enter invalid username or invalid password

- Click “Sign in” button

- Do steps on 3 and 4 on several times

- Verify that we get some new message beside message “Wrong usename or password” and banned for input new username and password

Source:

/testing/LoginTestCase/T6\_test.py

1. T7 - SQL Injection Security Testing

Steps:

- Navigate to url http://localhost/login.php

- Verify sign in page is visible

- Enter valid username and password

- Click “Sign in” button

- Verify that message “Wrong usename or password” appear on website

Source:

/testing/LoginTestCase/T6\_test.py

1. Create TestCase on feature:
2. LogoutTestcase

Steps:

- Navigate to url http://localhost/login.php

- Verify sign in page is visible

- Enter valid username and password

- Click “Sign in” button

- Verify that logged in into dashboard as username

- Click “Log out” button

- Verify that you are in login page

Source:

/testing/AnotherFeatureTestCase/LogoutTestcase\_test.py

1. CreateContactTestcase

Steps:

- Navigate to url http://localhost/login.php

- Verify sign in page is visible

- Enter valid username and password

- Click “Sign in” button

- Verify that logged in into dashboard as username

- Click “Create” button

- Verify that you are in create contact page

- Enter all valid data form

- Click “Submit” button

- Verify that you are in dashboard page

- Search name on search field

- Verify that name on table

Source:

/testing/AnotherFeatureTestCase/CreateContactTestcase\_test.py

1. EditContactTestcase

Steps:

- Navigate to url http://localhost/login.php

- Verify sign in page is visible

- Enter valid username and password

- Click “Sign in” button

- Verify that logged in into dashboard as username

- Click “Create” button

- Verify that you are in create contact page

- Enter all valid data form

- Click “Submit” button

- Verify that you are in dashboard page

- Search name on search field

- Verify that name on table

- Click “Edit” button

- Verify that you are in edit contact page

- Enter all valid new data form

- Click “Submit” button

- Verify that you are in dashboard page

- Search name on search field

- Verify that new data is updated on table

Source:

/testing/AnotherFeatureTestCase/EditContactTestcase\_test.py

1. DeleteContractTestcase

Steps:

- Navigate to url http://localhost/login.php

- Verify sign in page is visible

- Enter valid username and password

- Click “Sign in” button

- Verify that logged in into dashboard as username

- Click “Create” button

- Verify that you are in create contact page

- Enter all valid data form

- Click “Submit” button

- Verify that you are in dashboard page

- Search name on search field

- Verify that name on table

- Click “delete” button

- Click on “submit” button

- Verify that you are in dashboard page

- Search name on search field

- Verify that name is deleted on table

Source:

/testing/AnotherFeatureTestCase/DeleteContactTestcase\_test.py

1. ProfileTestcase

Steps:

- Navigate to url http://localhost/login.php

- Verify sign in page is visible

- Enter valid username and password

- Click “Sign in” button

- Verify that logged in into dashboard as username

- Click “Profile” button

- Verify that you are in profile page

- Upload new image

- Verify new image is uploaded

Source:

/testing/AnotherFeatureTestCase/ProfileTestcase\_test.py

1. XSSTestcase

Steps:

- Navigate to url http://localhost/login.php

- Verify sign in page is visible

- Enter valid username and password

- Click “Sign in” button

- Verify that logged in into dashboard as username

- Click “XSS” button

- Input XSS payload on field

- Click “Submit” button

- Verify XSS payload is running

Source:

/testing/AnotherFeatureTestCase/XSSTestcase\_test.py

1. ShowEntryTestcase

Steps:

- Navigate to url http://localhost/login.php

- Verify sign in page is visible

- Enter valid username and password

- Click “Sign in” button

- Verify that logged in into dashboard as username

- Click “Dropdown Entry” button

- Choose for max (100) entry

- Verify that data on table is show max 100 data

Source:

/testing/AnotherFeatureTestCase/ShowEntryTestcase\_test.py

Pipeline CI/CD:

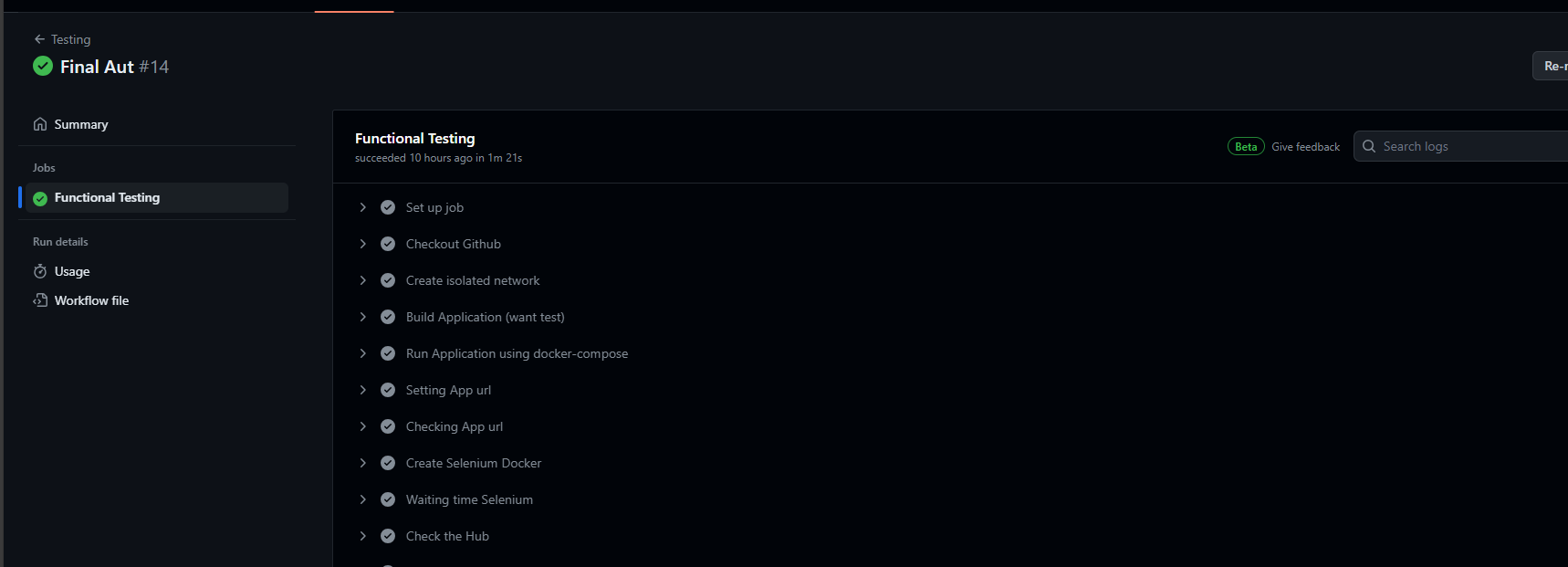
Steps:

* Github Checkout
* Create isolated network
* Build application
* Build Apache and mysql use docke-compose
* Setting URL application and validation
* Create Selenium docker and checking
* Installation Selenium Client on python
* Python Test script running

Source main.yml

|  |
| --- |
| name: Testing  on:    push:      branches:        - main  jobs:    test:      name: Functional Testing      permissions: write-all      runs-on: ubuntu-latest      steps:        - name: Checkout Github          uses: actions/checkout@v4        - name: Create isolated network          run: docker network create --driver bridge isolated        - name: Build Application (want test)          run: docker build -t my-php-site:latest .        - name: Run Application using docker-compose          run: docker-compose up -d          - name: Setting App url          run: |            URL=http://$(ip -f inet -o addr show docker0 | awk '{print $4}' | cut -d '/' -f 1)            echo "URL=$URL" >> $GITHUB\_ENV        - name: Checking App url          run: curl ${{ env.URL }}          - name: Create Selenium Docker          run: docker run --network=isolated -d -p 4444:4444 -p 7900:7900 -v /dev/shm:/dev/shm --name firefox-standalone selenium/standalone-firefox:latest        - name: Waiting time Selenium          run: sh wait.sh        - name: Check the Hub          run: curl http://localhost:4444/ui        - name: Installing Selenium client on python          run: pip install -U selenium pytest pytest-xdist        - name: Run python testing          run: |            pytest -v -n 5 testing/AnotherFeatureTestCase/        - name: stop docker          run: docker stop docker-apache-container |

Running test:



1. Dalam pengamanan SAST dan DAST pada aplikasi akan membuat file baru untuk iast.yml pada workflows.

Component yang digunakan:

1. ZAP for DAST

* ZAP is getting from https://github.com/hermanka/action-full-scan
* Source code configuration for ZAP on yml

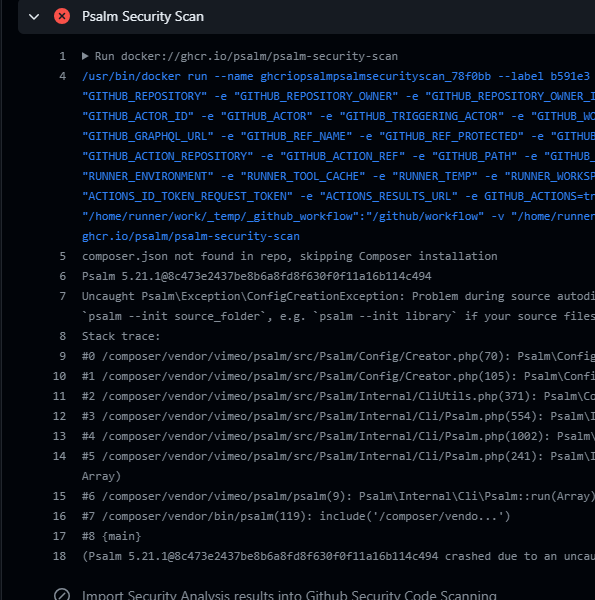
|  |
| --- |
| dast:      name: DAST      permissions: write-all      runs-on: ubuntu-latest      steps:        - name: Checkout the repo          uses: actions/checkout@v4        - name: Create isolated network          run: docker network create --driver bridge isolated        - name: Build AUT          run: docker build -t my-php-site:latest .        - name: Run AUT using docker-compose          run: docker-compose up -d          - name: Get AUT URL          run: |            URL=http://$(ip -f inet -o addr show docker0 | awk '{print $4}' | cut -d '/' -f 1)            echo "URL=$URL" >> $GITHUB\_ENV          - name: ZAP Scan          uses: hermanka/action-full-scan@master          with:            target: ${{ env.URL }}            network\_name: isolated        - name: stop docker          run: docker stop docker-apache-container |

1. Psalm for SAST

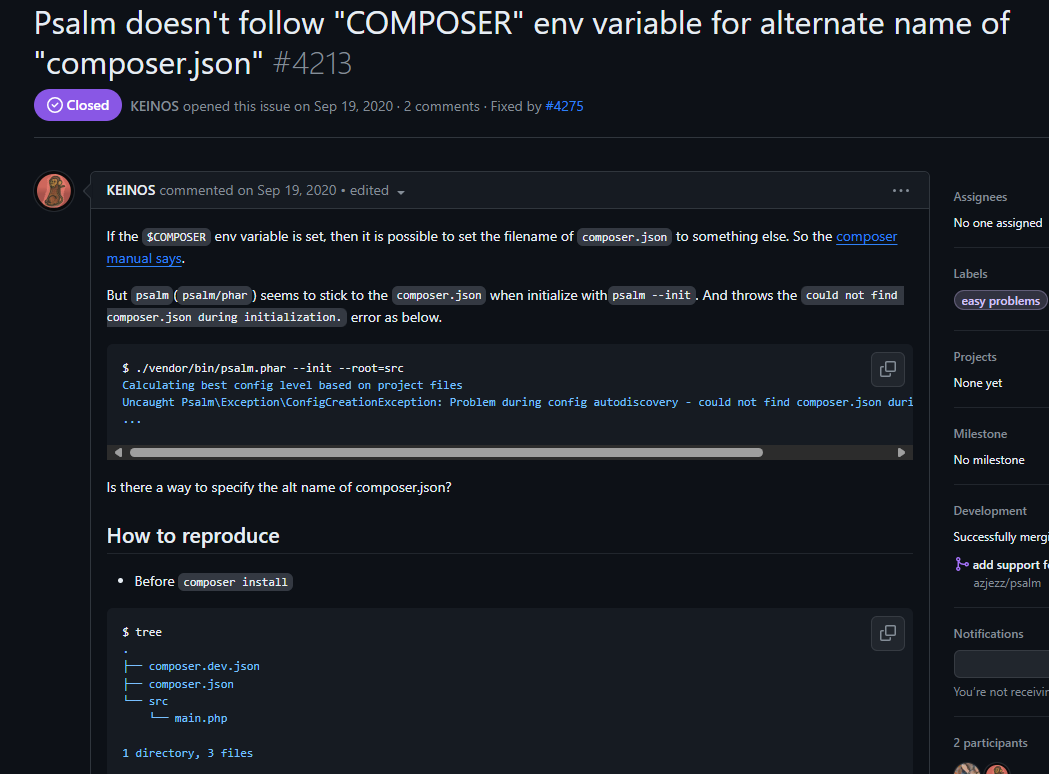
* Source code configuration for psalm on yml

|  |
| --- |
| sast:      name: Psalm      permissions: write-all      runs-on: ubuntu-latest      steps:        - name: Checkout the repo          uses: actions/checkout@v4        - name: Psalm Security Scan          uses: docker://ghcr.io/psalm/psalm-security-scan          with:            relative\_dir: ./src        - name: Import Security Analysis results into Github Security Code Scanning          uses: github/codeql-action/upload-sarif@v3          with:            sarif\_file: results.sarif |

* If you get this following error



* Solution is create new workshop tree and use src folder before make project



1. SonarCloud Scanning

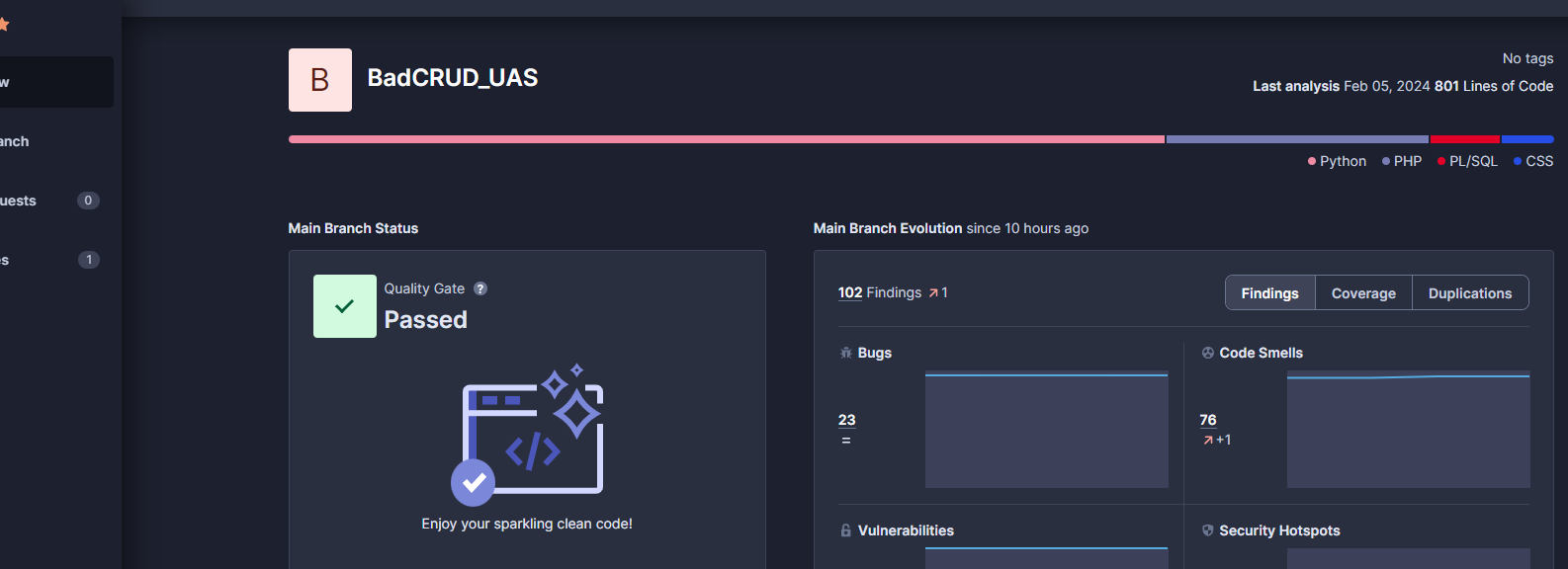
* Source code configuration for sonarcloud on yml

|  |
| --- |
| sonarcloud:      name: SonarCloud      runs-on: ubuntu-latest      steps:        - uses: actions/checkout@v3          with:            fetch-depth: 0        - name: SonarCloud Scan          uses: SonarSource/sonarcloud-github-action@master          env:            GITHUB\_TOKEN: ${{ secrets.GITHUB\_TOKEN            SONAR\_TOKEN: ${{ secrets.SONAR\_TOKEN }} |

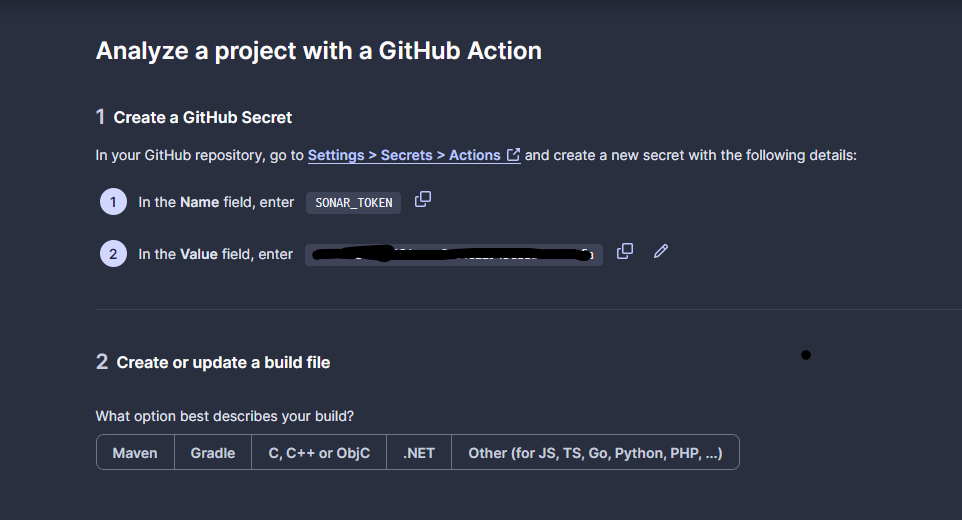
Also for sonar-project.properties

|  |
| --- |
| sonar.projectKey=fixxall\_BadCRUD\_UAS  sonar.organization=fixxall |

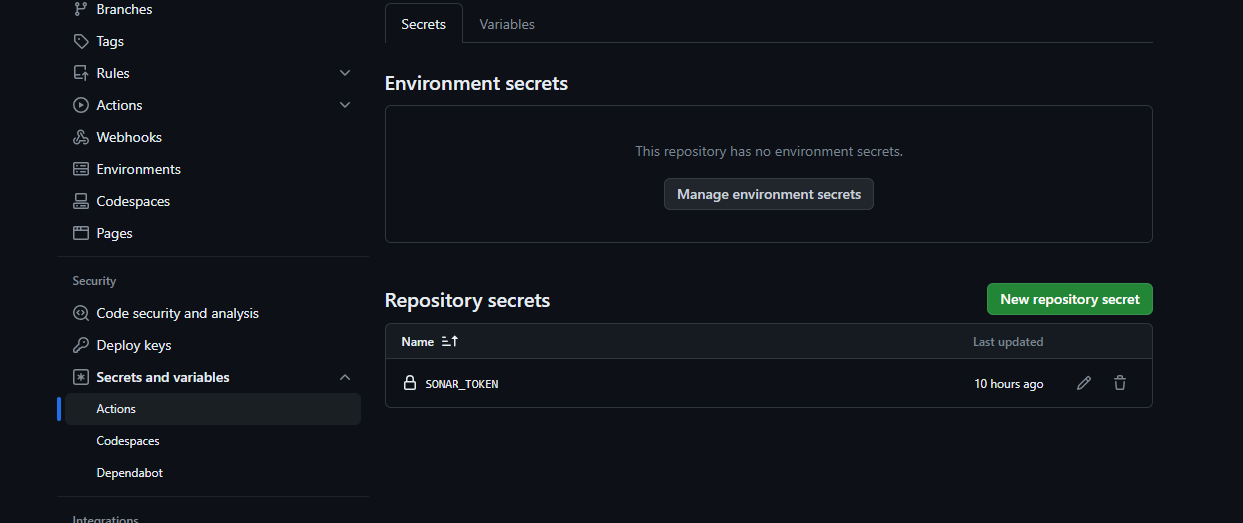
* Visit <https://sonarcloud.io/projects> and create project on my repository



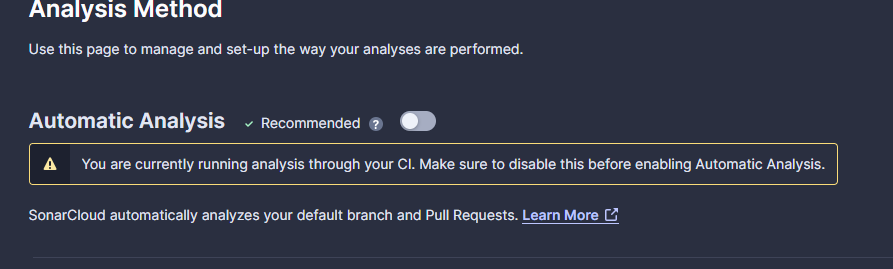
* Go on menu Administration > Analisys Method > with Github Methods and follow instruction



* Create secret actions on github repository

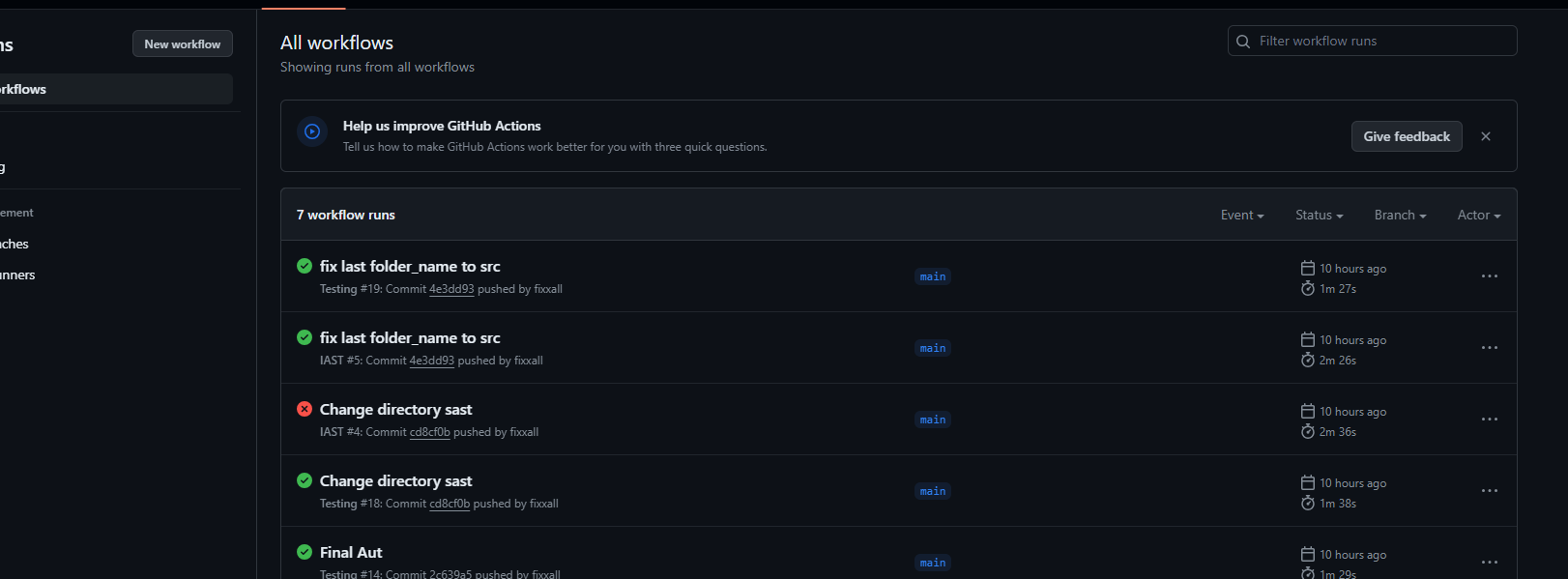


* If some error happen on running actions check for the analisys method is disable

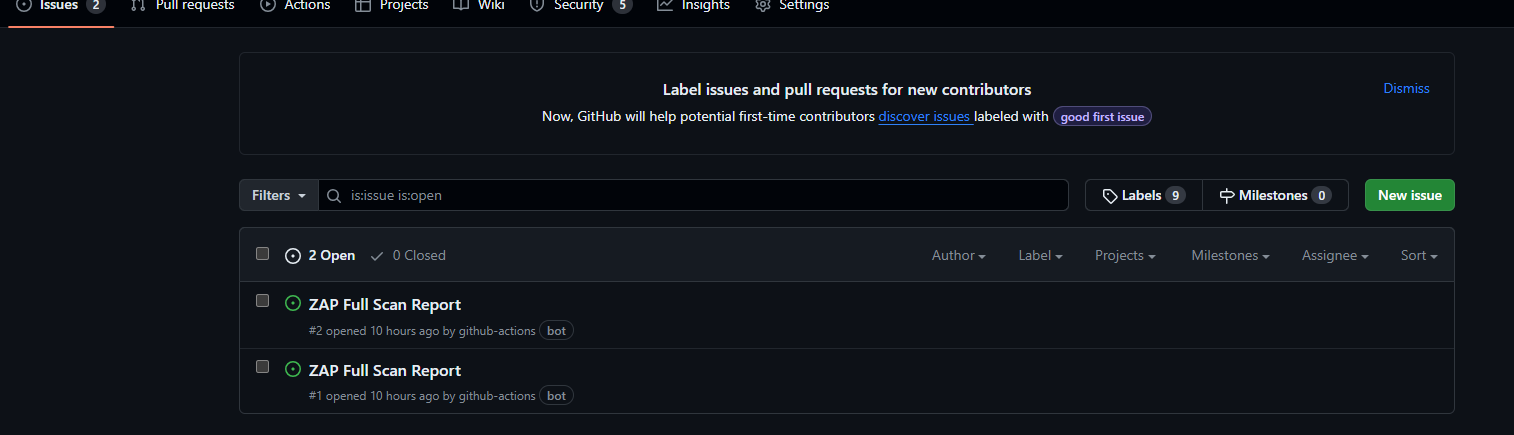


1. Result

* Running for actions can be found on actions menu



* ZAP report can viewed on issues



1. Menerapkan pengujian otomatis, termasuk Dynamic Application Security Testing (DAST), Static Application Security Testing (SAST), dan pengujian fungsional dalam CI/CD pipeline untuk pengembangan aplikasi Android merupakan langkah penting untuk memastikan keamanan dan kualitas perangkat lunak secara kontinu. Berikut adalah langkah-langkah umum yang dapat Anda ikuti:
2. Persiapkan Infrastruktur dan Lingkungan:

Pastikan Anda memiliki infrastruktur yang memadai untuk menjalankan pengujian otomatis. Ini mungkin melibatkan server pengujian, perangkat fisik atau emulator Android, dan perangkat lunak yang diperlukan. Atur emulator atau perangkat fisik yang diperlukan untuk menjalankan pengujian fungsional.

1. Gunakan Alat Pengujian Keamanan:
2. DAST (Dynamic Application Security Testing): Pilih alat DAST seperti OWASP ZAP atau Burp Suite. Konfigurasi alat untuk melakukan serangan keamanan dinamis terhadap aplikasi Android.
3. SAST (Static Application Security Testing): Pilih alat SAST seperti Checkmarx, Fortify, atau SonarQube. Integrasikan alat ini ke dalam tahap pengujian kode sumber pada pipeline CI/CD.
4. Pengujian Fungsional: Gunakan kerangka pengujian otomatis seperti Appium atau Espresso. Tulis skrip pengujian untuk menguji fungsionalitas aplikasi secara otomatis.
5. Integrasi Alat Keamanan ke dalam Pipeline CI/CD:
6. DAST: Tentukan tahap dalam pipeline CI/CD (misalnya, setelah pengujian unit atau integrasi) di mana DAST harus dijalankan. Konfigurasikan alat DAST untuk berkomunikasi dengan aplikasi yang sedang diuji.
7. SAST: Integrasikan alat SAST ke dalam tahap pengujian kode sumber, mungkin setelah pemeriksaan sintaks dan kompilasi. Pastikan hasil pengujian SAST terintegrasi dengan laporan CI/CD.
8. Pengujian Fungsional: Tentukan tahap di pipeline CI/CD (setelah DAST dan SAST, misalnya) di mana pengujian fungsional akan dijalankan. Sesuaikan skrip pengujian fungsional dengan pipeline CI/CD untuk eksekusi otomatis.
9. Pantau dan Evaluasi Hasil Pengujian:

Konfigurasi sistem untuk secara otomatis memonitor hasil pengujian setelah setiap iterasi. Bangun mekanisme pemberitahuan atau peringatan untuk masalah keamanan atau kegagalan pengujian fungsional.

1. Jalankan Pengujian Saat Deployment ke Lingkungan Pengujian/Staging:

Pastikan pengujian otomatis, termasuk DAST dan SAST, dijalankan sebagai bagian dari proses deployment ke lingkungan uji/staging sebelum merilis ke produksi.

1. Penanganan Hasil Pengujian:

Jika pengujian gagal, hentikan proses CI/CD dan berikan laporan detail mengenai masalah yang ditemukan. Pertimbangkan untuk mengotomatiskan pembuatan tiket/tugas untuk memperbaiki masalah yang ditemukan.

1. Evaluasi dan Pemeliharaan:

Terus pantau dan evaluasi efektivitas pengujian otomatis. Perbarui skrip pengujian, konfigurasi alat, dan strategi pengujian sesuai kebutuhan dan perubahan dalam aplikasi.

1. Keamanan dalam Proses Pengembangan:

Libatkan pengembang dalam siklus pengujian keamanan dan pastikan mereka memahami hasilnya.

Dengan menerapkan langkah-langkah ini, Anda dapat memastikan bahwa pengujian otomatis, termasuk keamanan (DAST dan SAST) dan pengujian fungsional, terintegrasi dengan lancar ke dalam pipeline CI/CD, membantu meminimalkan risiko keamanan dan memastikan kualitas aplikasi Android secara berkelanjutan.