Checkmarx

CxFlow

Training Exercises

21 March 2023

Table of Contents

Introduction

This document contains various CxFlow exercises.

This document assumes that the students have GitHub accounts (these can be created for the training and then abandoned).

CxFlow

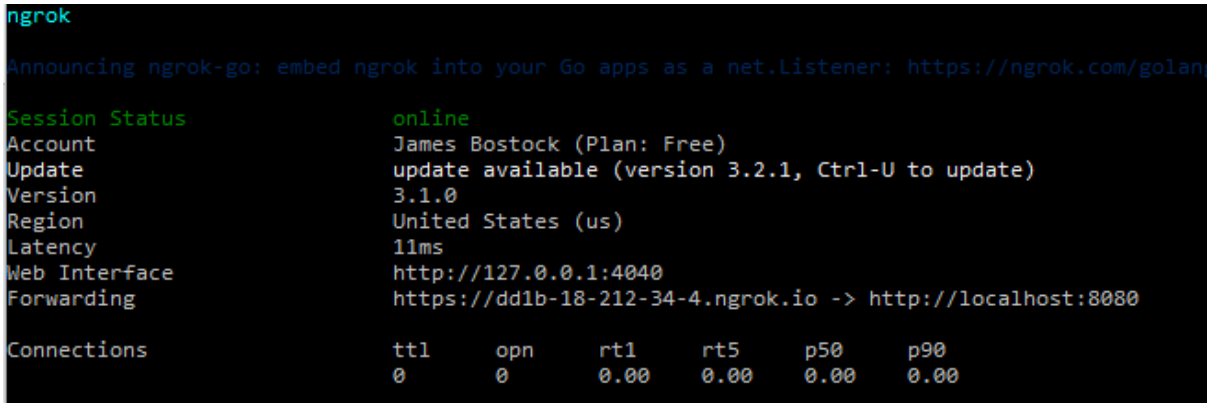
The training virtual machines come with an old version of CxFlow. Download the latest release from the CxFlow GitHub repository: https://github.com/checkmarx-ltd/cx-flow/releases. As the training virtual machines have Java 8 installed, download the Java 8 version of the jar file.

Ngrok

As a rule, the virtual machines used for the training exercises are not accessible from the public internet. The most pragmatic way to circumvent this limitation is to use ngrok, a free ingress-as-a-service platform. To use ngrok,:log in using GitHub as the authentication provider (alternatively, you can register and use email and password, or, if you have a Google account, you can also use Google as the authentication provider). You will be taken to a page of instructions:

1. Download the ngrok software.
2. Extract the contents of the zip archive.
3. Connect your account.
4. Start ngrok (which port to use depends on the exercise).

One cool thing about ngrok is that it starts a local web server which allows you to see the requests that are received and the responses that are sent.



CxFlow Webhook with GitHub

Start Ngrok

By default, in server mode, CxFlow listens on port 8080 so we start ngrok with the following command.

```
PS C:\Ngrok>.\ngrok http 8080
```

Create a GitHub Repository

Create a new repository in GitHub. In this document we will call it 'CxFlow-Webhook-Test', but you can give it any name you like.

Use the GitHub UI's editor to add a file. For example, `src/main.py`, with the following contents.

```
import os
import sys

if __name__ == '__main__':
    for arg in sys.argv[1:]:
        os.remove(arg)
```

Configure a Repository Webhook

Go to the repository's 'Settings' tab and select 'Webhooks'.

Click the 'Add Webhook' button.

Copy and paste the public Ngrok URL into the 'Payload URL' field.

Use the online UUID generator to generate a UUID and paste it into the Webhook's 'Secret' field[1].

Create the application.yml file

Use a text editor to create a file named `application.yml`, with the following content, in the same directory as the CxFlow jar file.

```
cxflow:
  bug-tracker: GitHub
  branches:
    - main

checkmarx:
  username: admin@cx
  password: H#c2iN36M!
  version: 9.0
  team: /CxServer
  base-url: http://localhost

github:
  webhook-token: <UUID>
  token: <Your GitHub API token>

logging:
  level:
```

[1] Note that the secret does not have to be a UUID – it can be any string. Using UUIDs for this use case is very common, though.

```
com:
  checkmarx: DEBUG
```

Start CxFlow

Open a PowerShell window and navigate to the folder containing the CxFlow jar file.

Run the following command:

```
PS C:\CxFlow> java -jar cx-flow-1.6.39.jar
```

Commit a Code Change

Make a change to the `main.py` file in the GitHub UI and commit the change.

CxFlow GitHub Action

Start Ngrok

The CxSAST instance on our training VMs listens on port 80 so we start ngrok with the following command:

```
PS C:\Ngrok> .\ngrok http 80
```

Create a GitHub Repository

Create a new repository in GitHub. In this document we will call it 'CxFlow-GitHubAction-Test', but you can give it any name you like.

Use the GitHub UI's editor to add a file. For example, `src/main.py`, with the following contents.

```
import os
import sys

if __name__ == '__main__':
    for arg in sys.argv[1:]:
        os.remove(arg)
```

Configure Repository Secrets

Go to the repository's 'Settings' tab and select the 'Secrets and variables' and then select 'Actions'.

Add the following secrets:

Secret Name	Secret Value
CHECKMARX_CLIENT_SECRET	
CHECKMARX_PASSWORD	The password of the user that CxFlow uses to access CxSAST.
CHECKMARX_PROJECT	The name of the CxSAST project.

Secret Name	Secret Value
CHECKMARX_TEAM	The team to which the CxSAST project belongs
CHECKMARX_URL	The base URL of the CxSAST instance.
CHECKMARX_USERNAME	The username of the user that CxFlow uses to access CxSAST.

Create a GitHub Workflow

Create a file named `.github/workflows/checkmarx.yml` with the following content.

```
# This workflow uses actions that are not certified by GitHub.
# They are provided by a third-party and are governed by
# separate terms of service, privacy policy, and support
# documentation.

# This is a basic workflow to help you get started with Using Checkmarx CxFlow Action

name: CxFlow

on:
  push:
    branches: [ main ]
  pull_request:
    # The branches below must be a subset of the branches above
    branches: [ main ]

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
# - this job is specifically configured to use the Checkmarx CxFlow Action
permissions:
  contents: read

jobs:
  # This workflow contains a single job called "build"
  build:
    # The type of runner that the job will run on - Ubuntu is required as Docker is
    # leveraged for the action
    permissions:
      contents: read # for actions/checkout to fetch code
      issues: write # for checkmarx-ts/checkmarx-cxflow-github-action to write feedback
                    # to github issues
      pull-requests: write # for checkmarx-ts/checkmarx-cxflow-github-action to write
                           #feedback to PR
```

```
    security-events: write # for github/codeql-action/upload-sarif to upload SARIF results
  runs-on: ubuntu-latest

  # Steps require - checkout code, run CxFlow Action, Upload SARIF report (optional)
  steps:
  # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
  - uses: actions/checkout@v3
  # Runs the Checkmarx Scan leveraging the latest version of CxFlow - REFER to Action
  # README for list of inputs
  - name: Checkmarx CxFlow Action
    uses: checkmarx-ts/checkmarx-cxflow-github-action@v1.5
    with:
      project: ${{ secrets.CHECKMARX_PROJECT }}
      team: ${{ secrets.CHECKMARX_TEAMS }}
      checkmarx_url: ${{ secrets.CHECKMARX_URL }}
      checkmarx_username: ${{ secrets.CHECKMARX_USERNAME }}
      checkmarx_password: ${{ secrets.CHECKMARX_PASSWORD }}
      checkmarx_client_secret: ${{ secrets.CHECKMARX_CLIENT_SECRET }}
      scanners: sast
      params: >-
        --namespace=${{ github.repository_owner }}
        --repo-name=${{ github.event.repository.name }}
        --branch=${{ github.ref }}
        --cx-flow.filterSeverity
        --cx-flow.filterCategory
        --logging.level.com.checkmarx=DEBUG
        --cx-flow.zip-exclude=\.git/.*,\\.github/.*
  # Upload the Report for CodeQL/Security Alerts
  - name: Upload SARIF file
    uses: github/codeql-action/upload-sarif@v2
    with:
      sarif_file: cx.sarif
```

Creating the file should trigger a scan.