National College of Ireland

Configuration Manual

MSc Research Project
Cyber Security

Arjun Variammattu Sasi
Student ID: x22180770

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: **Arjun Variammattu Sasi**

Student ID: **X22180770**

Programme: **MSc Cybersecurity** **Year:** **2024**

Module: **MSc Research Project**

Lecturer: **Michael Pantridge**

Submission Due Date: 25/04/2024

Project Title: **Enhancing Web App Security in CI/CD Pipeline: A DevSecOps Framework with Open-Source Tools**

Word Count: **1555** **Page Count: 10**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Arjun Variammattu Sasi
Date: 25/04/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Date:	
Penalty Applied (if applicable):	

Configuration Manual

Arjun Variammattu Sasi
Student ID: x22180770

1 Introduction

The configuration manual introduces a comprehensive DevSecOps framework tailored for web applications, addressing the critical imperative of seamlessly integrating security into CI/CD pipelines. This framework automates security testing processes, encompassing Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and manual penetration testing methodologies. Executed on AWS platforms via the GitLab CI/CD pipeline, it streamlines security assessments and enhances deployment efficiency. A pivotal feature of this framework is its ability to amalgamate tool outputs, enabling the generation of unified security audit reports. Empirical evaluations and case studies validate the effectiveness of this solution in fortifying the security posture of web applications. By presenting a comprehensive approach to security testing, grounded in practical tools and methodologies, this thesis contributes significantly to advancing DevSecOps practices, bridging crucial gaps in existing literature and offering tangible insights for developers and stakeholders alike.

Presentation URL: https://youtu.be/OtH4oUkOi-I

2 Hardware Requirements

Operating System: Windows 11
RAM: 8.0 GB
Processor: 11th Gen Intel Core i5-11320H @ 3.20GHz 2.50 GHz
Storage: 512 GB SSD
System Type: 64-bit operating system, x64-based processor

3 Software Requirements

To execute the specified GitLab CI/CD pipeline effectively, specific versions of the required software and tools are recommended based on the details provided in your configuration. Here are the tools along with the suggested versions or version requirements:

- **GitLab Runner**: Ensure it's compatible with your GitLab version. Generally, using the latest version is recommended.
- **Docker**: The pipeline uses images like ***docker:latest*** and ***docker:stable***. Ensure your Docker version supports these images, typically Docker 19.03 or newer.
- **Docker-in-Docker** (**DinD**): The service version used is specified in config as **docker:dind** and **docker:20.10.11-dind**, Docker 20.10.11 or a compatible version for DinD services.

- **Snyk**: Use the latest Docker image version of Snyk **(snyk/snyk:docker)** to ensure up-to-date vulnerability databases.
- **Trivy**: The pipeline dynamically fetches the latest version of Trivy. Ensure **wget** and **tar** are available to handle this operation.
- **OWASP ZAP**: The configuration does not specify a version, but using the latest stable Docker image (**softwaresecurityproject/zap-bare**) is advisable.
- **Nmap**: The pipeline uses **apk add nmap** without specifying a version, indicating the latest stable version available in the Alpine package repository is used.
- **Amazon AWS CLI**: Version 2.15.32 is explicitly used in the pipeline. Ensure this version or a newer one is used for compatibility with AWS services.
- **Node.js** and **NPM**: The pipeline specifies **node:lts-alpine3.19**, which refers to the Long-Term Support (LTS) version of Node.js available at the time of the **Alpine 3.19** release. It typically includes the latest LTS version of Node.js and npm.
- **Python**: Version **3.11.9** is used within an **Alpine 3.19** image (**python:3.11.9-alpine3.19**), so ensure this specific version or a newer compatible version is used for script execution.
- **GNU gettext** and **wget**: These tools are used implicitly without version specifications. The latest stable versions available in your environment's package manager should suffice.
- **xsltproc**: No version specified; use the latest available version that is compatible with your environment.

4 Configure AWS

4.1 Create AWS Service Role

- Navigate to the AWS Management Console and log in. From the dashboard, select "**Services**" and click on "**IAM**" under the "**Security, Identity, & Compliance**" section.
- Within the IAM dashboard, click "**Roles**", then "**Create role**".
- Choose "**AWS service**" as the trusted entity and select the specific service that will use this role.
- Attach the below listed existing policies to the role

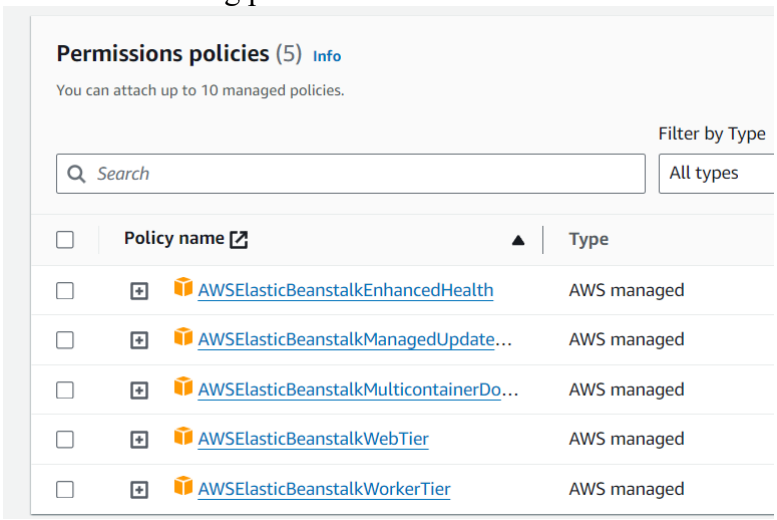

Figure 1:AWS Role Policies

- Give the role the name aws-elasticbeanstalk-service-role and description, then click “Create role”.

4.2 Create an Elastic Beanstalk Application

- In the AWS Management Console, find and open the **Elastic Beanstalk service** under the "**Compute**" section.
- Click **Create Application**.
- Enter an Application Name as ***juiceshop*** and, optionally, a Description. Click Create.
- Select the platform **docker** as application's environment.
- Review all settings and configurations and Click on **Create environment**
- Choose environment tier**: Web server environment**
- Give Environment name as ***juiceshop-env***
- Select Platform as **Docker**
- Choose **Sample application** for Application code
- In the Configure service access, select **Use an existing service role** and choose **aws-elasticbeanstalk-service-role** which was previously created
- **Skip to review**
- Click on **Create environment**.

5 Generate a Snyk Personal Access Token

- Log into Snyk account. If you don't have one, sign up at snyk.io
- Access the account settings by clicking on profile or the account icon.
- Navigate to API tokens. Look for a section related to API tokens or security settings within account or settings page.
- Generate a new token.

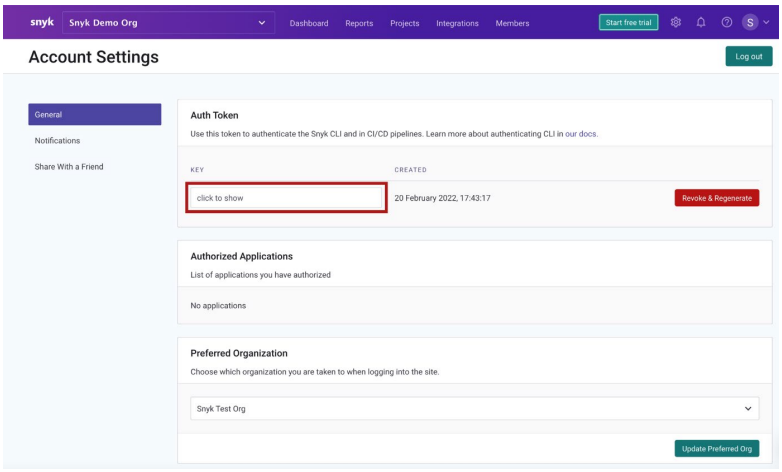

Figure 2: Generate Snyk PAT

- Copy the generated token.

6 Configuring Gitlab

6.1 Preparation of Files

Initially, the project files contained within **sentinalsecopsartifacts.zip**, including **.gitlab-ci.yml**, **Dockerfile**, and the contents of **scripts** and **templates** directories, need to be prepared for upload.

6.2 Accessing the GitLab Repository

The user must log into their GitLab account and navigate to the target project repository where the integration of the OWASP Juice Shop code and additional project files is intended.

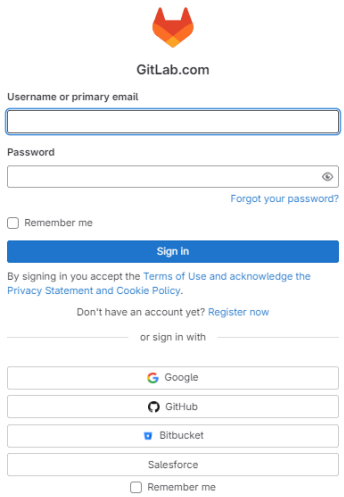

Figure 3: GitLab Login

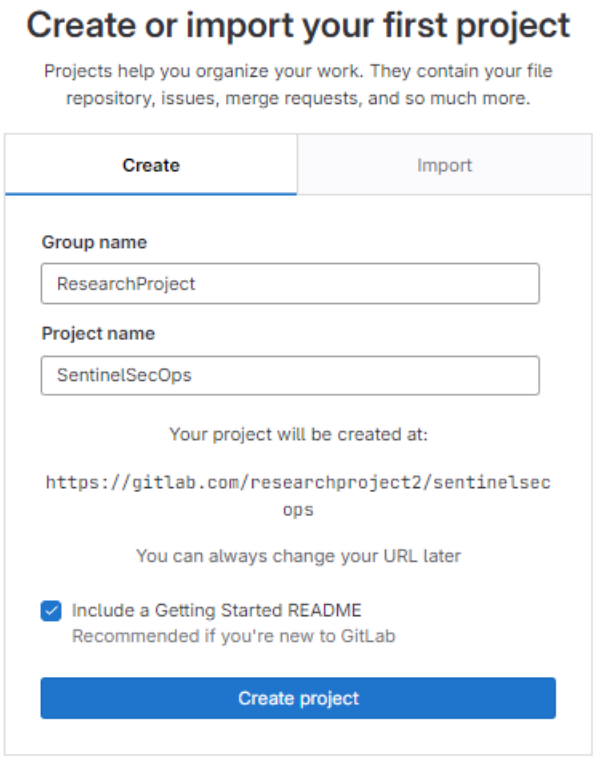

Figure 4: Create Project

6.3 Cloning OWASP Juice Shop Repository

- Visit the OWASP Juice Shop GitHub page https://github.com/juice-shop/juice-shop.
- The repository is to be downloaded as a ZIP file.

- Open the Gitlab Repository in WebIDE Editor

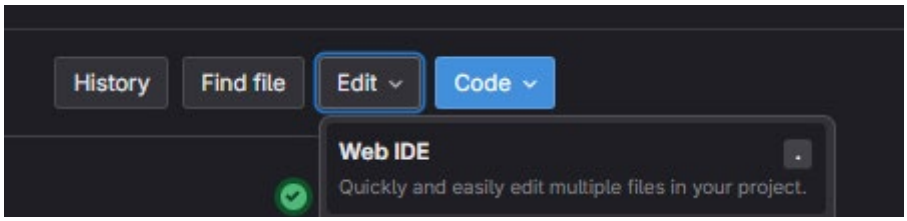

Figure 5: GitLab Web IDE

- Upload the Juice Shop files using the **WebIDE**.

6.4 Creating a Deploy Token for the repository

- Navigate to Your Repository
- Click on "Settings" and then select "Repository".
- Scroll to "Deploy Tokens" and click on "Expand"
- Fill out the form with the token's name as aws, optional expiry date, and scopes read_repository, read_registry.
- Click on "Create deploy token".
- Copy and securely store the displayed token

6.5 Adding variables to a GitLab CI/CD pipeline

- Go to your project's dashboard.
- Find the Settings: On the left-hand sidebar, expand the Settings menu by clicking on it.
- Access CI/CD Settings: Inside the settings menu, click on CI / CD to open the CI/CD settings page.

6.5.1 Add Variables

- Expand the **Variables** section. Scroll down to find the "**Variables**" section and click on "**Expand**".
- Add the Snyk token as a new variable:
  - Click on "**Add Variable**"
  - In the "**Key**" field, enter the name of variable.
  - Make it "**Protected**" (only available in protected branches or tags) and "**Masked**" (hidden in job logs).
- Click "**Add variable**" to save.

The following variables are to be added to Gitlab

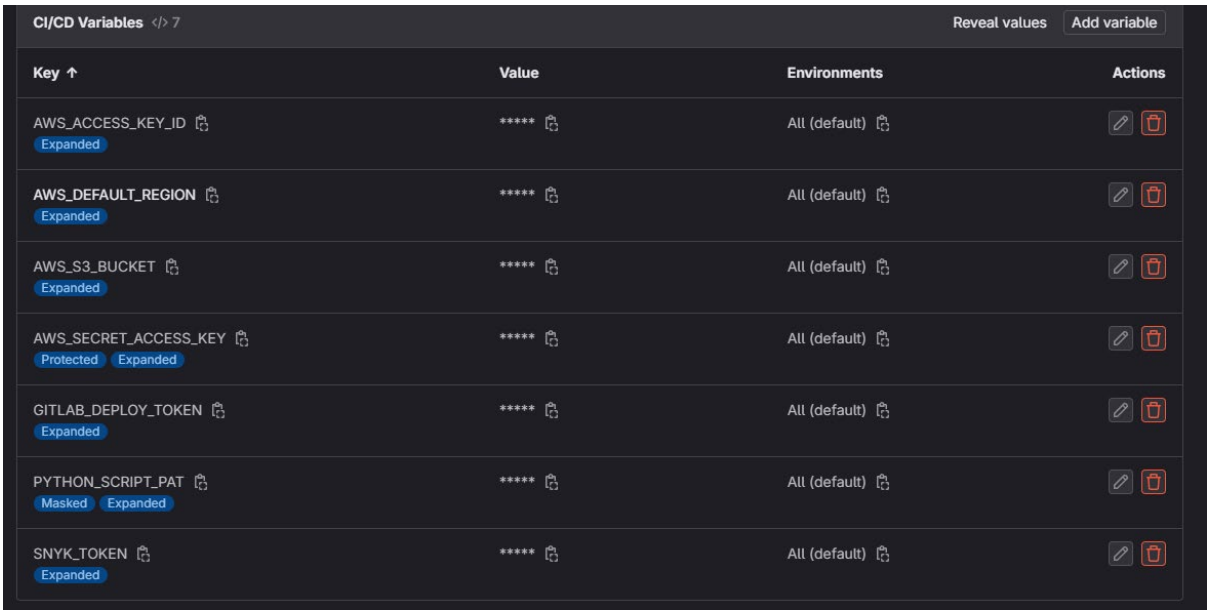

Figure 6: Variables

Variable	Description
AWS_ACCESS_KEY_ID	Access Key ID of User created in AWS IAM
AWS_DEFAULT_REGION	Default Region of AWS Services
AWS_S3_BUCKET	Name of the S3 bucket created by AWS EBS
AWS_SECRET_ACCESS_KEY	Access Key of User created in AWS IAM
PYTHON_SCRIPT_PAT	Previously created Gitlab Account PAT for the python script
GITLAB_DEPLOY_TOKEN	Previously created Gitlab Account PAT for aws deployment Value format: aws:<tocken>
SNYK_TOKEN	Token obtained from snyk.io
AWS_URL	URL of deployed AWS web application

Figure 2: Variables and description

6.6 Uploading Project Files

The ***sentinalsecopsartifacts.zip*** file needs to be extracted.

- Within the Web IDE, necessary directories (**scripts** and **templates**) are to be created by right-clicking in the file tree area and choosing new directory.
- Upload each project file to its corresponding directory by right-clicking the directory and selecting Upload file. This includes **.gitlab-ci.yml**, **Dockerfile**, **issue_creator.py**, **requirements.txt**, **auth.json**, **Dockerrun.aws.json**, and **Dockerrun.aws.public.json**.

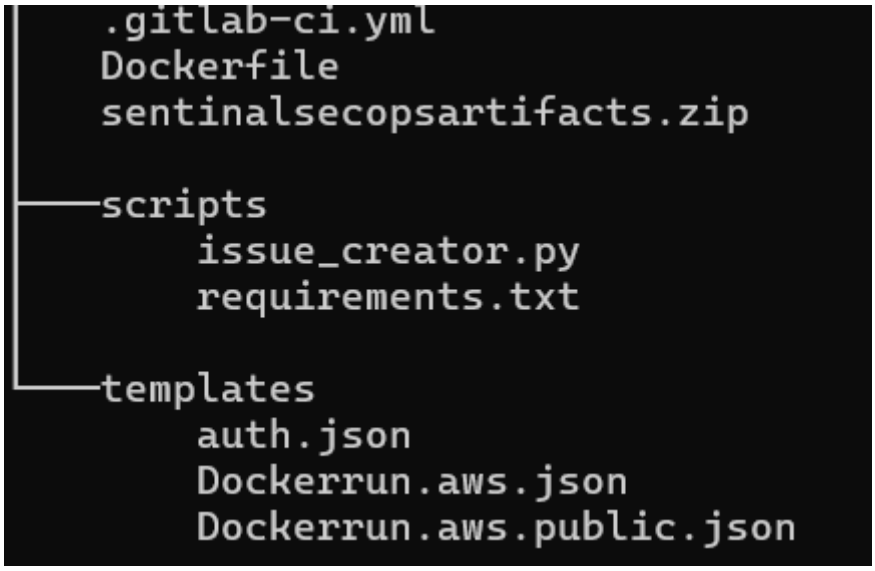

Figure 7: File structure

6.7 Committing Changes

- Once all files are uploaded, changes should be reviewed by the user in the Web IDE.
- To commit these changes, the Commit button located in the bottom left corner is clicked.
- Eenter a meaningful commit message, such as “Add OWASP Juice Shop and project files”, in the commit message box.
- The commit action is finalized by clicking **Commit**.

6.8 Verification of Files

Exiting the Web IDE, the user should return to the project’s repository view on GitLab to confirm that all uploaded files and directories are accurately reflected. Go to the Pipeline to see if all the stages are successfull

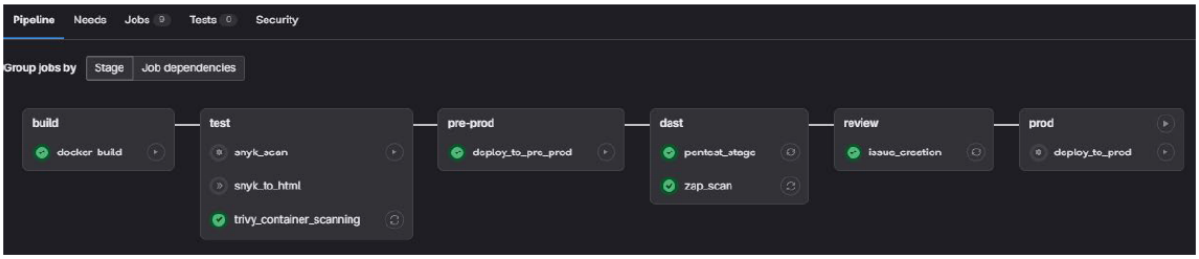

Figure 8: Pipeline