National College of Ireland

Configuration Manual

MSc Practicum Part 2
MSc Cybersecurity

Praveen Derenda Seetharam
Student ID: 23174501

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Praveen Derenda Seetharam
Student ID:	23174501
Programme:	MSc Cybersecurity **Year:** 2024-2025
Module:	MSc Practicum Part 2
Supervisor:	Vikas Sahni
Submission Due Date:	12/12/2024
Project Title:	Developing a Framework for Integrating Security Testing into the CI/CD Pipeline using Automation.
Word Count:	849 **Page Count** 6

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:	Praveen D S
Date:	11/12/2024

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):	

Configuration Manual

Praveen Derenda Seetharam
Student ID: 23174501

1 Introduction

This manual gives a step-by-step guide to configure and set up the OWASP Juice Shop web-application and use the secure CI/CD pipeline framework developed for integrating security testing into the CI/CD process. The project uses the following tools to ensure a secure development lifecycle:

- **GitHub Actions** -- https://github.com/features/actions
- **CodeQL** -- https://codeql.github.com/
- **Snyk** -- https://snyk.io/
- **OWASP ZAP** -- https://www.zaproxy.org/
- **Docker** -- https://www.docker.com/

2 System Configuration

• **Operating System**: macOS Sequoia
• **Hardware Configuration**:

- Chip: M3
- Memory: 16 GB
- Storage: 500 GB

• **Software Configuration**:

- **Docker**: Version 27.2.0, build 3ab4256
- **GitHub:** git version 2.44.0
- **Node.js**: Version v23.1.0
- **OWASP ZAP**: Version 2.15.0

3 Installation and Setup

OWASP Juice shop Setup:

To install and configure the OWASP Juice shop application on the local machine, the following are the requirements.

Prerequisites:

1. Install node.js[1] in your machine from the official website.

Installations:

The following are the steps for ease installation and deployment on localhost.
Step 1: Clone the git repository[2] into your machine from 1994dsp OWASP repository.
Step 2: Go into the cloned folder with cd juice-shop.
Step 3: Run npm install (must be done before first start or when you change the source code)
Step 4: Run npm start
Step 5: Browse to http://localhost:3000

[1] https://nodejs.org/en/download/package-manager
[2] https://github.com/1994dsp/Owasp

Installing Dependencies

Docker Installation:

Download Docker[3] desktop from Official Website and follow the installation instructions for your OS.

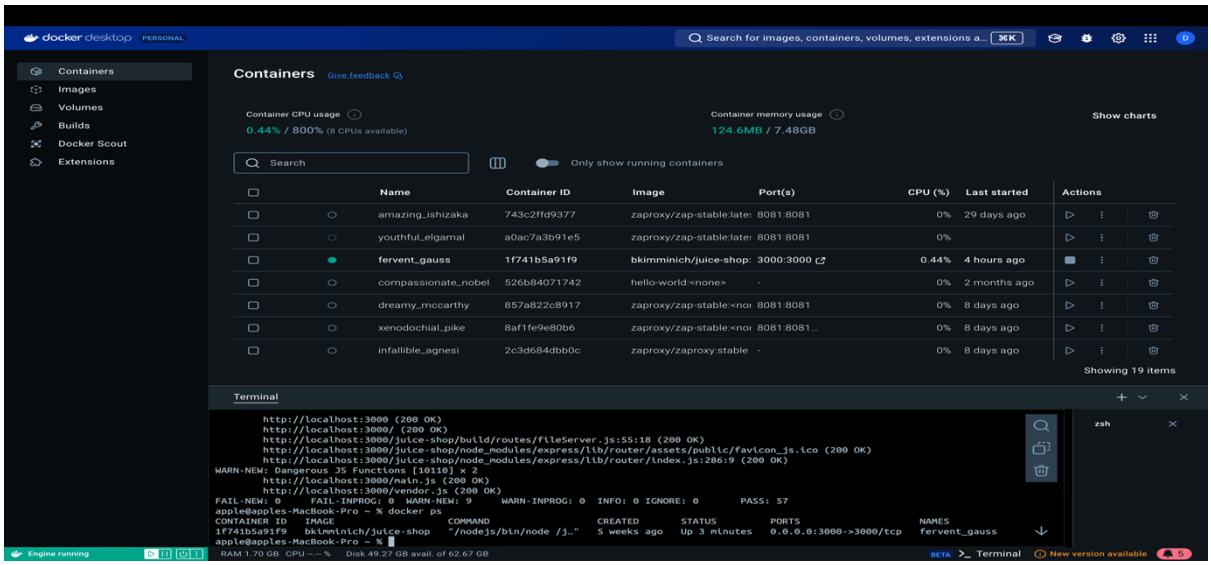

Figure 1:Docker Desktop for mac

4 Framework Configuration

GitHub Repository Configuration

1. Create a **GitHub Repository** and clone it to your local machine , here in this project I am using Visual Studio Code for the local development work.
2. Go to your project and Add a `.github/workflows/` directory to store CI/CD pipeline YAML files as shown in the below figure.

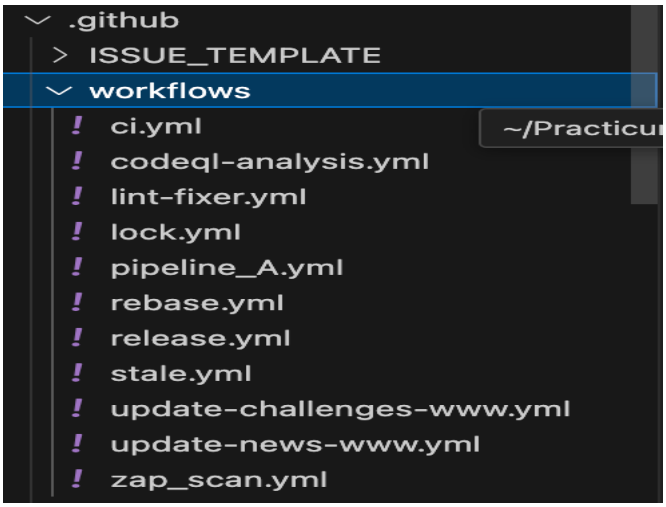

Figure 2:YML files under .github/workflows

[3] https://www.docker.com/

GitHub Actions Workflow

1. Create a `ci.yml` file in `.github/workflows/`.
2. Add configurations for:
    - **Code Linting**: Using ESLint
    - **Static Analysis**: Using CodeQL
    - **Dependency Scanning**: Using Snyk

```yaml
.github > workflows > ! lint-fixer.yml
name: "Let me lint:fix that for you"

on: [push]

jobs:
  LMLFTFY:
    runs-on: ubuntu-latest
    steps:
    - name: "Check out Git repository"
      uses: actions/checkout@3df4ab11eba7bda6032a0b82a6bb43b11571feac #v4.0.0
    - name: "Use Node.js 18"
      uses: actions/setup-node@5e21ff4d9bc1a8cf6de233a3057d20ec6b3fb69d #v3.8.1
      with:
        node-version: 18
    - name: "Install CLI tools"
      run: npm install -g @angular/cli
    - name: "Install application"
      run: |
        npm install --ignore-scripts
        cd frontend
        npm install --ignore-scripts --legacy-peer-deps
    - name: "Fix everything which can be fixed"
      run: 'npm run lint:fix'
    - uses: stefanzweifel/git-auto-commit-action@3ea6ae190baf489ba007f7c92608f33ce20ef04a #v4.16.0
      with:
        commit_message: "Auto-fix linting issues"
        branch: ${{ github.head_ref }}
        commit_options: '--signoff'
        commit_user_name: JuiceShopBot
        commit_user_email: 61591748+JuiceShopBot@users.noreply.github.com
        commit_author: JuiceShopBot <61591748+JuiceShopBot@users.noreply.github.com>
```

Figure 3:Lint config file

Security Tools Integration

Static Code Analysis (CodeQL):

- Enable CodeQL in your repository's Security Settings.
- Add CodeQL to your `ci.yml`

```yaml
.github > workflows > ! codeql-analysis.yml
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@3df4ab11eba7bda6032a0b82a6bb43b11571feac #v4.0.0
    - name: Initialize CodeQL
      uses: github/codeql-action/init@v2
      with:
        languages: ${{ matrix.language }}
        queries: security-extended
        config: |
          paths-ignore:
            - 'data/static/codefixes'
    - name: Autobuild
      uses: github/codeql-action/autobuild@v2
    - name: Perform CodeQL Analysis
      uses: github/codeql-action/analyze@v2
```

Figure 4:CodeQl.yml file

Dependency Scanning (Snyk):
Add Snyk extension into your Visual studio Code

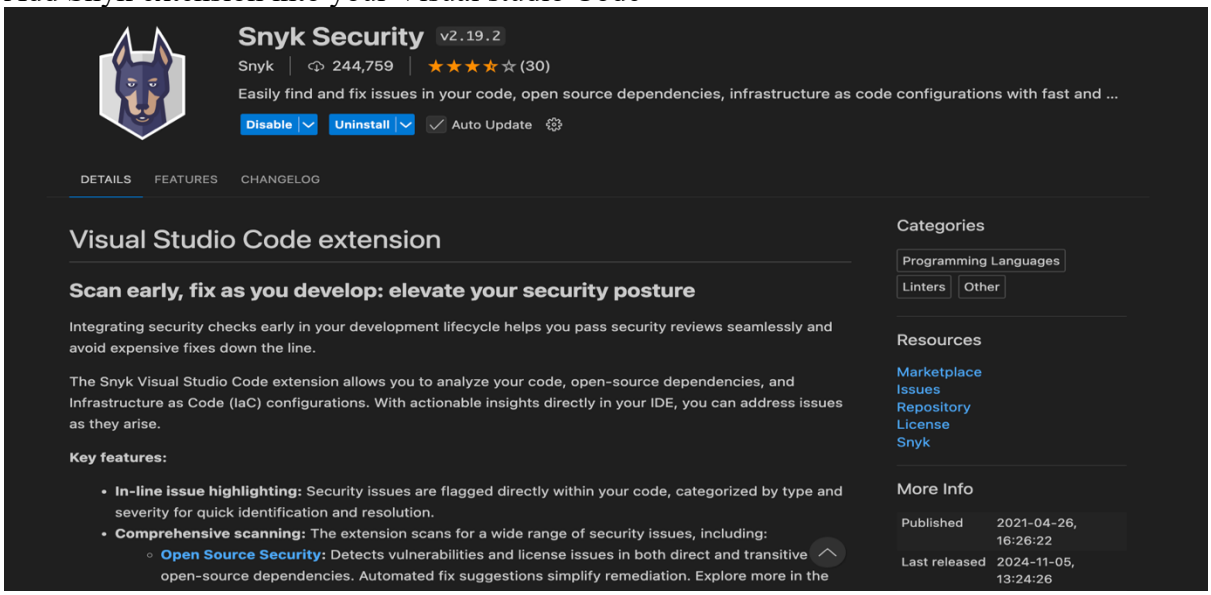

Figure 5:Snyk extension

Dynamic Security Testing (OWASP ZAP):

Add OWASP ZAP to your `ci.yml`:

```
- name: "Run Juice Shop Application"
  run: |
    docker run -d --name juice-shop-container -p 3000:3000 dsp1994/owasp:snapshot
```

5 Usage Instructions

1. **Push Changes**:
   - Commit and push code to the repository:
     ```
     git add .
     git commit -m "Initial CI/CD pipeline setup"
     git push origin main
     ```

2. **Monitor Workflow**:
   - Navigate to the **Actions** tab in GitHub to view pipeline progress.

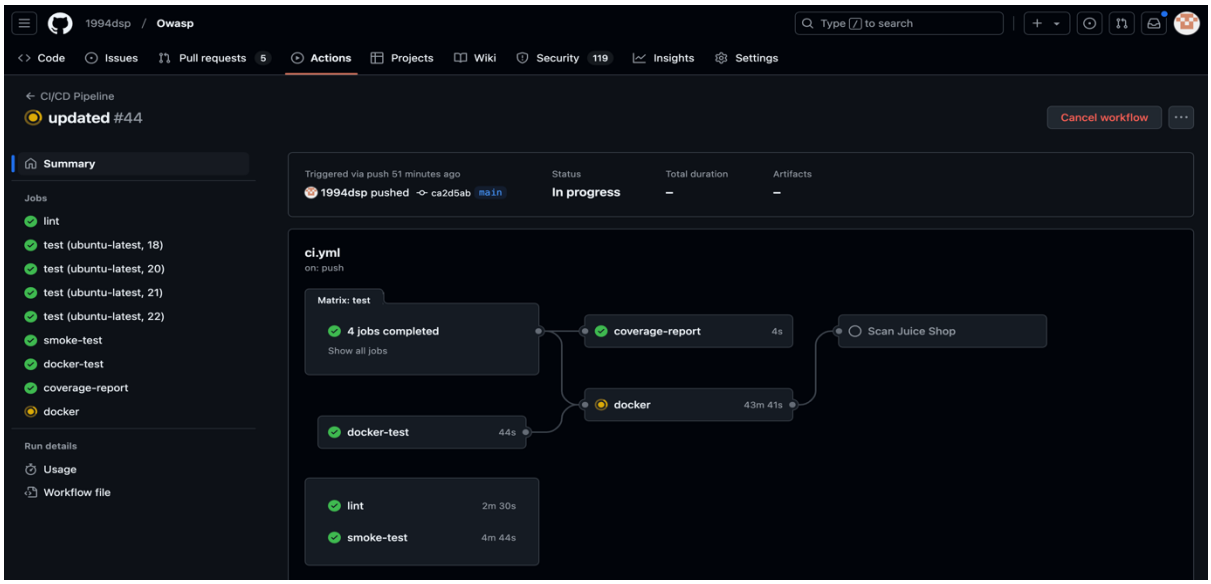

Figure 6: CICD workflow

3. **Generate Reports**:
    - Review logs and reports generated by tools like CodeQL, Snyk, and OWASP ZAP.

6 Validation and Testing

1. **Validate Static Analysis**:
    - Push insecure code (e.g., with SQL injection) and ensure CodeQL flags it.

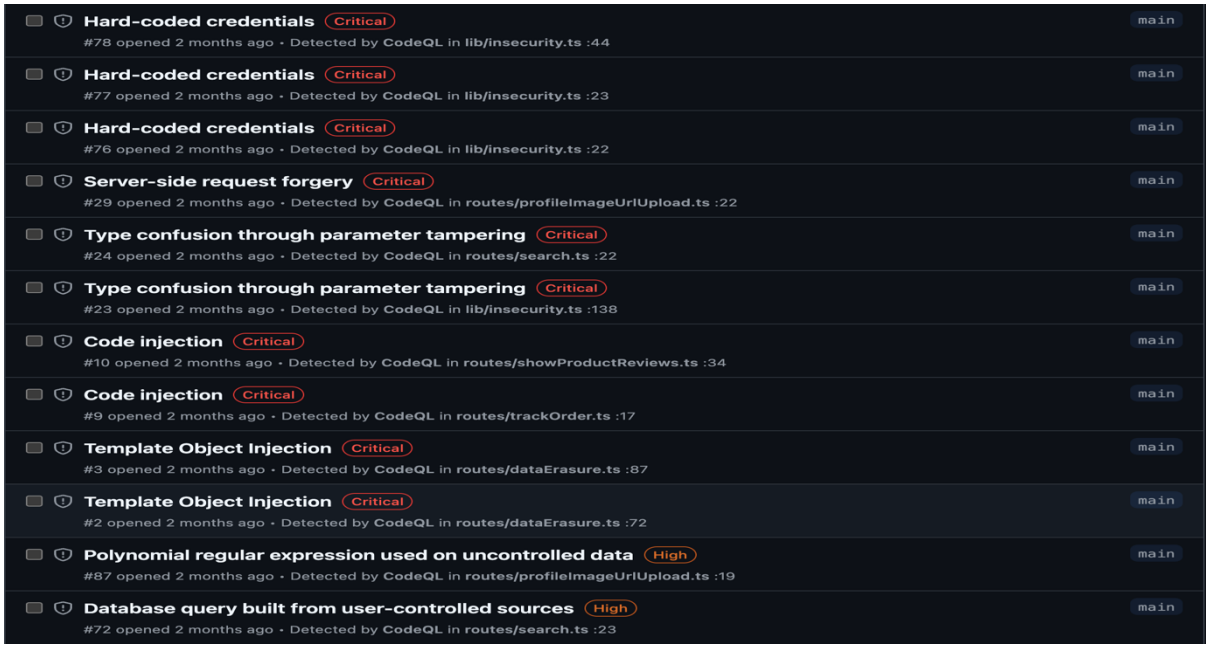

Figure 7:CodeQl report

2. **Run Dependency Scans**:
    - Use outdated libraries in `package.json` to validate Snyk scans.

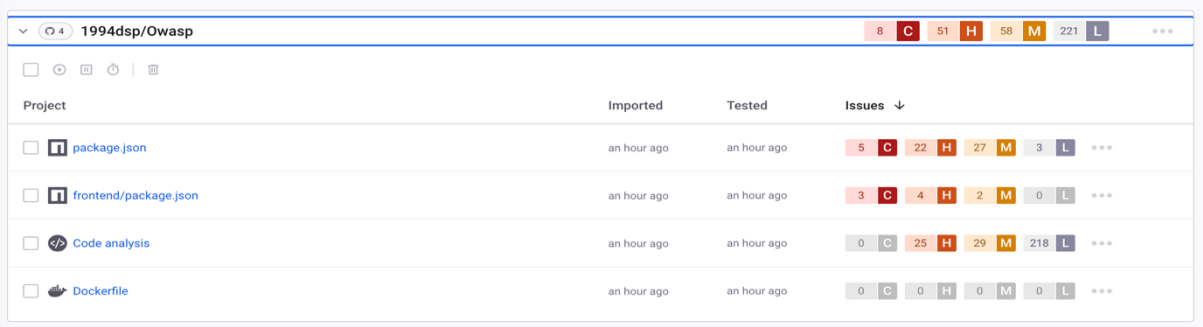

Figure 8:Snyk report

3. **Test Runtime Security**:
    - Deploy OWASP Juice Shop and run OWASP ZAP against it.

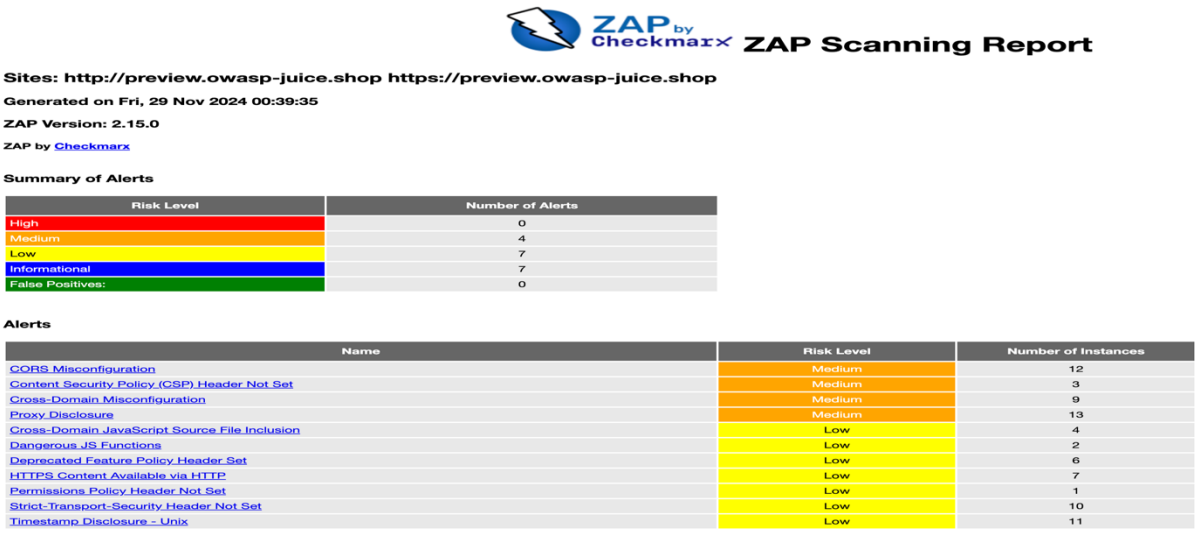

Figure 9: ZAP report.

7 Conclusion

This manual equips developers with the knowledge to configure a secure CI/CD pipeline. By automating testing and integrating robust tools, this framework ensures efficient and secure software delivery.

References

Decan, A., Mens, T., Mazrae, P.R. and Golzadeh, M., 2022, October. On the Use of GitHub Actions in Software Development Repositories. In 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 235-245). IEEE.