OT Lab3 (DevSecOps)

Web applications are playing a vital part in today's digital world. Knowing how to find the security issues laying inside your web application before its deployment and finding its vulnerabilities before others after deployment would be the majority of your tasks as a Security Specialist.

Moreover, almost all companies use microservice architecture as well as DevOps methodology. Therefore in this lab, you will learn how to integrate security tools into your CI/CD pipeline and continuously build, security test, and deploy the secure software. In simple words, how to implement the DevSecOps CI/CD pipeline.

Task 1: Set up your environment.

1. Create two VMs, for the Jenkins server and its agent.
    a. Install Docker Engine on both VMs.
2. Isolate your Jenkins server behind a reverse proxy and access it through port 80 (e.g Nginx Reverse Proxy).
3. You will be working on the following Maven project. Go through it and familiarize yourself (git clone https://github.com/HamidullahMuslih/ot-lab-sdlc.git).

    Note: the project is written in Springboot (Backend) and Bootstrap (Frontend) and already contains unit test cases.
4. Read and explain in one line sentence the Maven lifecycle. Understand commands for each phase (e.g build, unit test, integration test, packaging and etc.) since you will need to use them during the pipeline tasks.

Task 2: Implement DevSecOps Pipeline

Create the below Jobs in the same sequence:

1. Create the first job to fetch and build the Maven project. Note: skip the testing in this job.
2. Create the next job to perform the unit tests of the project.
3. Create the next job to perform the integration test on the project.
4. Create the next job to perform static code analysis on the project (e.g with the CheckStyle, PMD, FindBugs tools). Note: Read about Warnings Next Generation Plugin.
    a. Show the reports and share your understanding.

   b. Feed the static code analysis reports to the [Violations Plugin] and set the quality gates for the number of bugs in one of the reports, validate your work to fail/unstable the job. Note: after validation remove the quality gate restriction since it will fail the last task of the lab.
5. Create the next job to dockerize your artifact from target/****.war and push it to the docker hub.
6. Create the next job to deploy the docker image from the docker hub to the Jenkins agent and validate/show that you can access the web app.
7. Finally, chain all the jobs such that when you run the 1st job it should execute the rest. Note: Use Post-Build Actions>Build other projects.

Task 3: (Optional) I need more

1. Integrate Slack notification such that for each job the developer gets notified whether the job is a succus, failed, unstable. (Easy)
2. Integrate Snyk with Jinkens and find project vulnerabilities.
3. Integrate Burbsuit in your pipeline and perform vulnerability scanning against the web application.
4. Deploy the SonarQube server in a VM or Docker container and perform static code analysis and set quality gates with SonarQube.
5. Write fuzz test cases and perform fuzzing on the software.
6. Deploy SonaType Nexus artifact server such that, after code analysis (6th job) the artifact should be uploaded to Nexus server.
7. Deploy K8s cluster and make master node as Jenkins agent.
   a. Write manifest files for the docker image.
   b. Create a job (or replacement of number eighth job) to deploy the app on K8s cluster.