Docker Security Vulnerabilities – Gradle Plugin

Paul Kenny | X00107570

USE CASES

Use Case 1:

Title: Run Docker Scan

Primary Actor: Developer

Scope: Jenkins Pipeline

Level: Primary

1. Developer initiates the Jenkins pipeline to deploy a new component

1. The Jenkins pipeline builds, tests and analyses the component.
2. The Jenkins pipeline also builds a new Docker image containing the developer's code changes.
3. The Gradle Security Vulnerabilities plugin scans the Docker image for known Java platform security vulnerabilities.
4. Any security vulnerabilities found are relayed to the developer through a security vulnerabilities report in browser interface.

When a developer is finished work on a new component a Jenkins pipeline is initialised. The pipeline will build the component using the component's Gradle build files. The component is then tested and analysed. The next step in the pipeline is to build a Docker image with the changes for deployment. Once the Docker image is complete the Gradle Security Vulnerabilities plugin will scan the Docker image and check against the Common Vulnerabilities and Exposures (CVE) database for known security vulnerabilities. A report will be compiled by the plugin which will be accessible to the developer through a browser interface.

TECHNICAL ARCHITECTURE

Software Components:

Git:

Git is an open source distributed version control system. It records changes to files over time. Specific versions can be recalled at any time and compared to previous or subsequent versions. Git will be used specifically in this project to track development changes throughout the development process.

Jenkins:

Jenkins is an open source automation server which automates parts of the software development process. Jenkins will be used to automate the component deployment pipeline. The pipeline will take a sample component, build it according to its Gradle build file instructions, test it, analyse it and then build a Docker image containing the component.

HTML:

Hypertext Mark-up Language is the standard mark-up language for creating web pages. HTML will be used to create a web interface which will display the results of Docker image vulnerability scans.

Java Archive Files:

A Java Archive file (JAR files) is a package file format used to aggregate Java class files and associated resources and meta data into one distributed application software. JAR files with known security vulnerabilities will be put into Docker images to be used in the development and testing stage of the Docker Security Vulnerabilities plugin.

Database:

In this project the Common Vulnerabilities and Exposures (CVE) list will be used to help identify security vulnerabilities in the scanned Docker images. CVE is a list of common names of publicly known security issues. The CVE list contains unique identifiers of these vulnerabilities and exposures. These identifiers enable the user to obtain information from a variety of CVE-Compatible information sources. CVE is sponsored by US-CERT, in the office of Cybersecurity and Communications at the U.S. Department of Homeland Security. (US-CERT, 2016) US-CERT advocates the use of CVE to the U.S. Government. The Docker Security Vulnerabilities plugin will access the database to check if a scanned Docker image contains any matching vulnerabilities.

Platform Libraries:

Gradle:

Gradle is a build tool and dependency manager for programming projects. Build scripts are written, in a domain specific language based on Groovy, which tell Gradle how to build an application. In this project the Docker Security Vulnerabilities scanner will be built as a Gradle plugin.

Groovy:

Groovy is an object-oriented programming language for the Java platform. The Docker Security Vulnerabilities plugin will be written in Groovy.

Docker:

Docker is an open-source project that automates the deployment of applications inside software containers. These containers wrap the application in a file system that contains everything needed to run the application consistently, regardless of the environment it will be running in. In this project, sample applications will be added to Docker images for deployment. The Docker images will be scanned by the Docker Security Vulnerabilities scanner to check for known security vulnerabilities.

Distribution and Development:

For the development stage of the project a distributed architecture will be used.

Developer:

The project developer will write and test the application code locally on their personal workstation. During the prototyping phase of the project, a sample application (containing JAR files with known vulnerabilities) will be added to a base Docker image. This image will be scanned by the prototype application for known security vulnerabilities. During the scanning process the application will cross-check with the Common Vulnerabilities and Exposures (CVE) list to check if a scanned Docker image contains any matching vulnerabilities. Once the scanning process is finished, scan findings will be reported to the developer through a web interface.

All application code will be pushed to IBM's GitHub and a virtual machine will be hosted by Amazon Web Services. This virtual machine will be used to run the prototyping process. The project developer's mentor at IBM will be able to access the application source code via GitHub and see the current prototype iteration in action in the virtual machine. This will facilitate continuous collaboration between the developer and mentor. (figure 1)

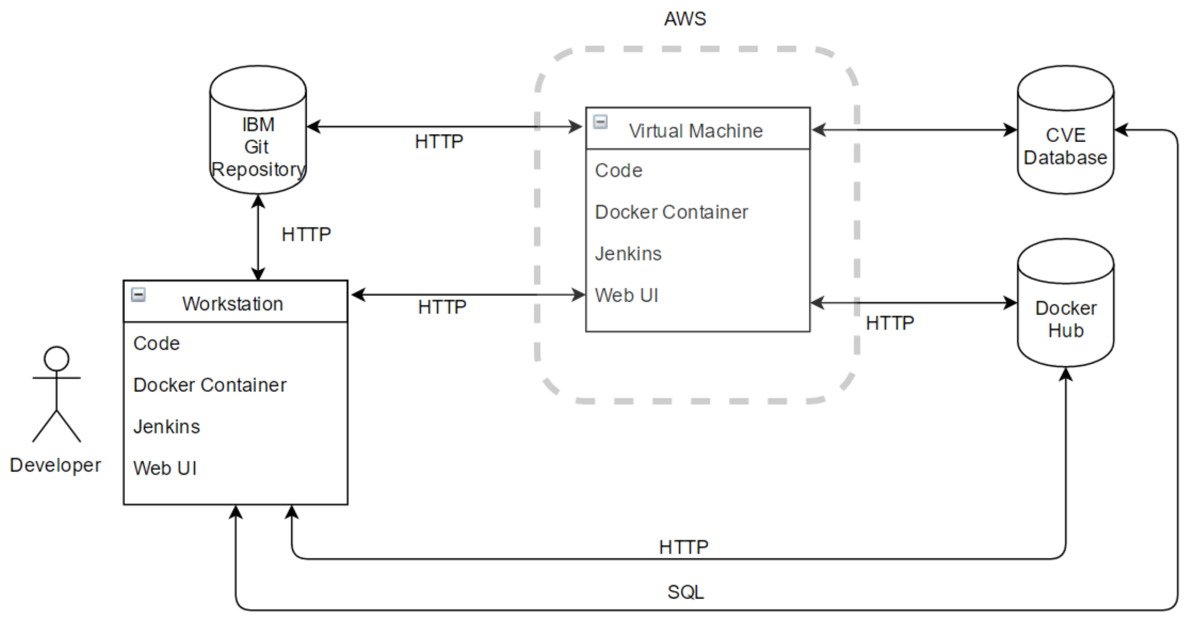

Figure 1: The development process distributed architecture

Database:

> The project will be utilising the Common Vulnerabilities and Exposures (CVE) database. The CVE is a dictionary of common names of known software security vulnerabilities. This project will cross-reference between scanned Docker containers and the CVE to check for security vulnerabilities.

Docker Hub:

> Docker Hub is a Docker image repository stored in the cloud. Images can be pushed to and pulled from the repository. This project will use Docker Hub to store Docker images which will be used in the prototype testing phase of the development process.

Virtual Machine (VM):

> A VM is an emulated computer system. Based on physical computer architecture, it provides all the same functionality as a physical system. In this project a VM will be used to facilitate continuous collaboration between the developer and mentor.

Git Repository:

> As stated above, Git will be used as the version control system to manage this project. The Git Repository is the location where all versions of the code in development will be stored. This project will use IBM's Git repository.

Workstation:

> For this project the primary workstation will be the laptop provided to the developer from IBM. The laptop will communicate with IBM through a VPN provided by IBM.

Security Certificates, Authentication:

> Both the developer and IBM mentor will share access to the GitHub repository and the Amazon Web Services Virtual Machine. Access to IBM's GitHub repository is secured through the developer and mentor's IBM ID. Access to the AWS VM is secured through the AWS security credentials.

Risks:

The main risk involved in this project is creating a robust security vulnerability identification process. The system will only be as reliable as the vulnerability databases that are accessed to identify the vulnerabilities. If a new security issue arises and is not identified by the primary resource, then the target Docker container could be left exposed to attack. With this in mind it is proposed that a second phase of the project be considered. This phase will concentrate on potential future vulnerabilities. The commencement of the second phase will depend entirely on the completion of this project. It is felt that completing this project would be more beneficial to IBM in the time available.

An additional risk may arise during the scanning process. If, for unforeseen reasons, connection to the CVE is unavailable then the critical vulnerability cross-referencing process will not be able to complete. An appropriate way to mitigate this risk may be to migrate the CVE database to a new database in the cloud. This new database could be compared to the CVE and updated regularly. The initial overhead of migrating the CVE database may be large but the overhead associated with the maintenance of the migrated database would not be. However, database failover may fall outside the scope of the project and this issue may warrant further discussion.

Prototype:

The first prototype will consist of a simple Gradle plugin which will be able to scan a Docker image and identify simple files.

References:

US-CERT (2016) United States computer emergency readiness team. Available at: https://www.us-cert.gov/ (Accessed: 15 November 2016).