Project Research Document

**IBM Gradle Plugin for Security Vulnerabilities**

X00107570 | Paul Kenny

October 12, 2016

**Project Overview**

This project is to be undertaken in association with IBM. IBM developers utilise Gradle build-scripts to build software projects. These projects are then run in Docker containers which allow the applications to be deployed and tested in consistent environments.

When Gradle build-scripts are run, dependencies for these projects are specified within the build-scripts. These dependencies are usually third party, open source software that will allow the project to access and execute specific functionality provided by the dependencies. However, because the dependencies are from third party libraries and open source they may have previously unaccounted for security vulnerabilities which will in turn leave the project, and Docker image using them, vulnerable.

The objective of this project is to scan the Docker images and detect any security vulnerabilities which may be caused by the dependencies. This will require the development of a Gradle plugin which will check the dependencies of these projects for security vulnerabilities and notify the developers of potential problems which may arise.

This Gradle plugin will be written in Apache Groovy. It will access the dependencies specified in the Gradle build-scripts. These dependencies can then be checked to see if there are publicly disclosed vulnerabilities associated with them. For example, they will be cross-referenced against vulnerability databases like National Vulnerability Database (NVD, 2016), SourceClear (SourceClear, no date) or Common Vulnerabilities and Exposure (CVE, 1999) for up to date associated security vulnerabilities.

The plugin will download these databases/CSV files at regular specified intervals (maybe daily depending on the overhead associated with the size of the downloads) so the checks are always current. Once vulnerabilities are identified a report will be automatically generated and made available to the developer, who can then act on these vulnerabilities before the application is deployed. Alternatively, a Dashboard could be created to give the developer quickly digestible information provided by the report, which then could be investigated in further detail by the developer.

**Users**

The main users of the Gradle plugin for dependency security vulnerabilities will be IBM’s application developers and testers. However, the plugin will run every time the application is launched on a Docker deployment ensuring that the dependencies are always monitored for changes in the security status.

**Existing Applications**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Similarities | Differences |
| OWASP Dependency Check | Open-source tool which identifies project dependencies with known vulnerabilities. | Uses Gradle plugin to scan dependencies for vulnerabilities. | Receives vulnerability information from one source which may not be a comprehensive enough search. |
| RetireJS | Open-source JavaScript-specific dependency checker. | Retrieves vulnerability information from multiple resources. | No Gradle plugin and used on JavaScript specific projects. |
| SRC:CLR | Commercial dependency checker. | Retrieves vulnerability information from multiple resources and compiles to its own database. Provides Gradle support. | No Gradle specific plugin. |

# Platforms, Technologies and 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. Gradle creates a project object for each “build.gradle” script in the application. These objects allow the developer to access Gradle features.

The Gradle features fall into three categories: Properties, Tasks and Dependencies.

Properties consists of two property types, System Properties (configure the JVM that supports the execution of the build) and Project Properties (parameterise variables in the project).

Tasks are discrete pieces of work which tell Gradle how to build the project. Some tasks may have dependencies on other tasks, so the order in which they are executed is also controlled by the build script.

The final piece of the Gradle puzzle is the dependency. Dependencies are pieces of software, stored in repositories, that a project relies on to execute properly. Dependencies for the project and the repositories in which they are stored are specified in the build scripts to be managed by Gradle. This project’s focus will be on these dependencies and their security vulnerabilities.

**Apache Groovy:** Groovy is an object-oriented programming language for the Java platform. Gradle build-scripts, plugins etc. are written in Groovy. This is the main language the project will be written in.

**Docker Container:** Docker containers wrap a piece of software in a complete file system. This contains the environment needed to run the software. This basically guarantees that the software will always run the same regardless of the system it is running on. IBM use Docker to run their software, this project will do the same.

**External Vulnerability Databases:** This project will use external vulnerability databases to identify the current vulnerabilities of third party software which the targeted build is dependent on.

**GitHub:** Web based Git repository hosting service. It offers distributed version control and source code management. All code written for the project will be stored on GitHub.

**Risks**

The main risk involved in this project is creating a robust dependency security vulnerability identification process. Because the dependencies of the target build are predominantly third party open-source software located in third-party libraries and repositories, the opportunity for unidentified security issues to arise is great. The system will only be as reliable as the vulnerability databases that are accessed to identify the vulnerabilities. If a new security issue associated with a dependency arises and is not identified by one of these resources, then the target build could be left exposed to attack. Therefore, it is critical that as many resources as possible are identified and used for the security vulnerability checks. This may have a considerable overhead which will only be quantified once the project gets underway.

NVD (2016) National Vulnerability Database. Available at: https://web.nvd.nist.gov/view/vuln/search (Accessed: 12 October 2016).

SourceClear (no date) SourceClear. Available at: https://www.sourceclear.com/registry/explore (Accessed: 12 October 2016).

CVE (1999) Common vulnerabilities and exposures (CVE). Available at: https://cve.mitre.org/ (Accessed: 12 October 2016).