BLACKDUCK | Hub

Scanning Best Practices

Version 4.4.0

This edition of the *Scanning Best Practices* refers to version 4.4.0 of the Black Duck Hub.

This document created or updated on Monday, November 20, 2017.

Please send your comments and suggestions to:

Black Duck Software, Incorporated
800 District Avenue, Suite 201
Burlington, MA 01803-5061 USA

Contents

Chapter 1: Best Practices 1
- Android applications 2
- C and C++ 3
- C Sharp 5
- General support for all languages 6
- Java 7
- JavaScript 9
- Linux distributions 10
- Objective C 11
- PHP 12
- Python 13
- Ruby 14
- Scala 15
- Swift 16

The Hub documentation

The documentation for the Hub consists of online help and these documents:

Title	File	Description
Release Notes	release_notes_bd_hub.pdf	Contains information about the new and improved features, resolved issues, and known issues in the current and previous releases.
Installing Hub using Docker Compose	hub_install_compose.pdf	Contains information about installing and upgrading the Hub using Docker Compose.
Installing Hub using Docker Swarm	hub_install_swarm.pdf	Contains information about installing and upgrading the Hub using Docker Swarm.
Installing Hub using Kubernetes	hub_install_kubernetes.pdf	Contains information about installing and upgrading the Hub using Kubernetes.
Installing Hub using OpenShift	hub_install_openshift.pdf	Contains information about installing and upgrading the Hub using OpenShift.
Getting Started	hub_getting_started.pdf	Provides first-time users with information on using the Hub.
Scanning Best Practices	hub_scanning_best_practices.pdf	Provides best practices for scanning.
Getting Started with the Hub SDK	getting_started_hub_sdk.pdf	Contains overview information and a sample use case.
Report Database	report_db_bd_hub.pdf	Contains information on using the report database.

Hub integration documentation can be found on Confluence.

Training

Black Duck Academy is a one-stop resource for all your Black Duck education needs. It provides you with 24x7 access to online training courses and how-to videos.

New videos and courses are added monthly.

At Black Duck Academy, you can:

- Learn at your own pace.
- Review courses as often as you wish.
- Take assessments to test your skills.
- Print certificates of completion to showcase your accomplishments.

Learn more at https://www.blackducksoftware.com/services/training

View the full catalog of courses and try some free courses at https://academy.blackducksoftware.com

When you are ready to learn, log in or sign up for an account: https://academy.blackducksoftware.com

Customer Success Community

The Black Duck Customer Success Community is our primary online resource for customer support, solutions and information. The Customer Success Community allows users to quickly and easily open support cases and monitor progress, learn important product information, search a knowledgebase, and gain insights from other Black Duck customers. The many features included in the Customer Success Community center around the following collaborative actions:

- Connect – Open support cases and monitor their progress, as well as, monitor issues that require Engineering or Product Management assistance
- Learn – Insights and best practices from other Black Duck product users to allow you to learn valuable lessons from a diverse group of industry leading companies. In addition, the Customer Hub puts all the latest product news and updates from Black Duck at your fingertips, helping you to better utilize our products and services to maximize the value of open source within your organization.
- Solve – Quickly and easily get the answers you're seeking with the access to rich content and product knowledge from Black Duck experts and our Knowledgebase.
- Share – Collaborate and connect with Black Duck staff and other customers to crowdsource solutions and share your thoughts on product direction.

Access the Customer Success Community. If you do not have an account or have trouble accessing the system, please send an email to communityfeedback@blackducksoftware.com or call us at +1 781.891.5100 ext. 5.

To see all the ways you can interact with Black Duck Support, visit: https://www.blackducksoftware.com/support/contact-support.

Chapter 1: Best Practices

The Hub has general support for all languages. Below are links to information and best practices for the following:

- Android applications
- C and C++
- C#
- Java
- JavaScript
- Linux distributions
- Objective C
- Python
- Ruby
- Scala
- Swift

As an update to the discovery best practices described here, Black Duck recommends using Hub Detect, a command line interface (CLI) that integrates with your build jobs to identify package manager dependencies as well as file system matches. Hub Detect consolidates the functionality of the Black Duck build tools, package managers, and continuous integration plugin tools into a single tool. Hub Detect makes it easier to set up and scan code-bases using a variety of languages/package managers. When run against built binaries, the match accuracy of Hub Detect is very high. False positives and misses are greatly reduced.

Hub Detect supports the following package managers and languages:

Package Manager	Language	Forge	Required Files
NuGet 2	C#	NuGet.org	packages.config and (*proj or sln)
NuGet 3	C#	NuGet.org	project.lock.json or project.json and (*proj or *sln)
NuGet 4	C#	NuGet.org	project.assets.json or PackageReference in *proj
Vndr	GoLang	GitHub	vendor.conf

Package Manager	Language	Forge	Required Files
GoDep	GoLang	GitHub	godeps.json
Dep	GoLang	GitHub	gopkg.toml
Maven	Java	Maven Central	pom.xml
Gradle	Java	Maven Central	build.gradle
NPM	Node JS	npmjs	package.json
CocoaPods	Objective C	Cocoapods.org	podfile.lock
cpanm	Perl	CPAN	cpanfile
Conda	Python	Anaconda	environment.yml
Pear	PHP	Pear	package.xml
Composer	PHP	Packagist	composer.lock.composer.json
pip	Python	PyPi	setup.py
Packrat	R	CRAN	packrat.lock
RubyGems	Ruby	RubyGems	gemfile.lock
SBT	Scala	Maven Central	build.sbt

Click here for more information on Hub Detect.

Android applications

Category	Description
Summary Description	Android applications are matched using standard structure matching.
Discovery Best Practices	Black Duck recommends scanning the build inputs. This includes the source code which is being compiled for the application as well as any third party binaries or libraries that are included with the application. If the third party libraries are not standard binary distributions as provided from common download sites or if those binaries are created internally by compiling the source distributions, Black Duck recommends that the original source for those projects are scanned as well. As structure signature matching is the primary method that is used to find open source, Black Duck recommends scanning the open source in as close to the original structure as it would have been downloaded as possible.
Signature Scanning Source Code	Structure matching on file system directories and archives.

Category	Description
Signature Scanning Binaries	Special processing for *.dex files (Dalvik Executable)
Signature Scanning Archives	Typically, *.apk files (zip format)
Signature Scanning Qualifier	All matches are to collections such as directories, sub-directories, and archives, including unmodified and modified files. Typically, the files are contained in .apk archives which are expanded. There is special processing on .dex files to map those to java classes and improve matching results.
Build/Package Analysis	N/A
Declared Package Inspection	N/A
Primary KB Signature Content Sources	Java locations.

C and C++

Category	Description
Summary Description	The Hub provides limited support for C/C++. The Hub Scanner scans and identifies directories and sub-directories of source files and unmodified individual binaries of C/C++ projects from debian.org, openhub.net, and nuget.org. These represent the most popular C/C++ open source projects.
Discovery Best Practices	Black Duck recommends scanning the build inputs. This includes the source code which is being compiled for the application as well as any third party binaries or libraries that are included with the application. If the third party libraries are not standard binary distributions as provided from common download sites or if those binaries are created internally by compiling the source distributions, Black Duck recommends that the original source for those projects are scanned as well. As structure signature matching is the primary method that is used to find open source, Black Duck recommends scanning the open source in as close to the original structure as it would have been downloaded as possible.
Signature Scanning Source Code	Structure matching on file system directories and archives. Matches found via structure matching on source directories (such as *.c and *.h).

Category	Description
Signature Scanning Binaries	SHA1 matching on the following cross language binary types and object code: .dll, .exe, .so, and .o files. Matches to unmodified and modified collections of binary code re-used as provided by the open source project.
Signature Scanning Archives	C static libraries (*.a files) containing object code (*.o files) are expanded and scanned.
Signature Scanning Qualifier	All matches are to unmodified and modified collections such as directories, sub-directories, archives. • No matches to individual source files. • No matches to modified binaries. • Potentially could match modified static libraries (*.a files)
Build/Package Analysis	N/A
Declared Package Inspection	N/A
Primary KB Signature Content Sources	• C/C++ projects on OpenHub.net • C/C++ Linux distro source packages • C/C++ binaries from nuget.org

C Sharp

Category	Description
Summary Description	The Hub Scanner scans and identifies directories and sub-directories of source code and unmodified binaries from NuGet.org. The Hub-Nuget Plugin is available on GitHub. This plugin provides the ability to generate a Black Duck I/O formatted file containing the dependency information gathered from the C# projects. The file is generated in either the specified folder or defaults to the blackduck folder at the root of the solution or project. This plugin also has the ability to upload the Black Duck I/O file up to the Hub to create a scan file in the Hub. To generate the file and upload the contents to the Hub the buildBom.exe in the NuGet package for this integration must be executed in the directory containing the solution.
Discovery Best Practices	Black Duck recommends scanning the build inputs. This includes the source code which is being compiled for the application as well as any third party binaries or libraries that are included with the application. If the third party libraries are not standard binary distributions as provided from common download sites or if those binaries are created internally by compiling the source distributions, Black Duck recommends that the original source for those projects are scanned as well. As structure signature matching is the primary method that is used to find open source, Black Duck recommends scanning the open source in as close to the original structure as it would have been downloaded as possible.
Signature Scanning Source Code	Directories and sub-directories of .cs files.
Signature Scanning Binaries	Exact matches to individual .exe, .dll, .o, and .so files. Matches to unmodified and modified collections of binary code re-used as provided by the open source project.
Signature Scanning Archives	N/A
Signature Scanning Qualifier	All matches are to unmodified and modified collections such as directories, sub-directories, and archives, including NuGet packages. • No match to individual source files. • No match to modified binaries.
Build/Package Analysis	Hub-Nuget Plugin. The Hub-NuGet library provides an executable to gather NuGet project dependencies. This project will produce Black Duck I/O and has comparable functionality to the Hub-Maven and Hub-Gradle plugins.

Category	Description
	For use with the .NET framework.
Declared Package Inspection	Currently being investigated.
Primary KB Signature Content Sources	nuget.org
IDE Integration	Hub-Visual Studio plugin (also available on GitHub). The Black Duck Hub extension for Visual Studio uses your Black Duck Hub instance to provide an overview of NuGet packages in your solution.

General support for all languages

Category	Description
Summary Description	Signature-based scanning can be used for all languages supported by the Black Duck Hub.
Discovery Best Practices	Black Duck recommends scanning the build inputs. This includes the source code which is being compiled for the application as well as any third party binaries or libraries that are included with the application. If the third party libraries are not standard binary distributions as provided from common download sites or if those binaries are created internally by compiling the source distributions, Black Duck recommends that the original source for those projects be scanned as well. As structure signature matching is the primary method that is used to find open source, Black Duck recommends scanning the open source in as close to the original structure as it would have been downloaded as possible.
Signature Scanning Source Code	Structure matching on file system directories and archives.
Signature Scanning Binaries	SHA1 matching on the following cross language binary types and object code: .dll, .exe, .so, and .o files.
Signature Scanning Archives	Expansion of defined archive and compressed file types.
Signature Scanning Qualifier	N/A
Build/Package Analysis	Specific to individual package managers.
Declared Package Inspection	Specific to individual package managers.
Primary KB Signature Content Sources	Github, Google Code, Source Forge, Codeplex and many "long-tail" type repositories are the primary "multi-language" forges that provide content.

Java

Category	Description
Summary Description	The Black Duck Hub scans and identifies directories and sub-directories of source code and binaries from Maven Central and Apache projects (the majority of Java OSS is from these two locations). The Hub does not currently match individual java source files. Plugins are also available for Maven and Gradle. These plugins analyze resolved dependencies at build time adding to BOM accuracy. You can find these plugins on GitHub.
Discovery Best Practices	Identification of Java archives is very good through scanning only, however the addition of the Maven and Gradle plugins can increase the overall accuracy of discovery. To achieve the best results, implement the Maven or Gradle plugin to perform a build analysis over the dependencies, and then implement a scan over the build inputs (with the source checked out from the SCM) and the build outputs (.jar/.war/.ear files that result from the build). This allows the Hub to detect any open source that has been introduced outside of package management and corroborate those results against the declared packages. Of special interest will be those components only detected by package management and those only detected at the file level. If you are using Java without the Maven or Gradle plugins, make sure to scan both your build inputs (source checked out from SCM) and whatever build outputs are produced from Apache Ant or other build scripts. Component scanning will then include any open source introduced directly into the SCM and any third party artifacts (.jar files) that are introduced.
Signature Scanning Source Code	Structure matching on file system directories and archives.
Signature Scanning Binaries	SHA1 matching on the following cross language binary types and object code: .dll, .exe, .so, and .o files.
Signature Scanning Archives	Typically, jar, war, and ear files.
Signature Scanning Qualifier	All matches are to collections such as directories, sub-directories, and archives, including unmodified and modified jar files. No matches to individual .java or .class files.
Build/Package Analysis	Maven/Gradle
Declared Package Inspection	Potential future investigation.
Primary KB Signature Content Sources	Maven Central (maven.org) Apache projects from apache.org including source archives and Linux distributions.

Category	Description
IDE Integration	Hub-Eclipse Plugin is available on GitHub. This plugin uses your Black Duck Hub instance to provide an overview of all dependencies in your Maven and Gradle projects. The Hub-Eclipse plugin initially displays basic information, but also enables you to further drill down by opening the specific component version in the Black Duck Hub. The Black Duck Hub provides a rich set of metadata about components and releases to give developers clear insights to help them make informed decisions when selecting components for use. Features include the ability to: • Capture declared and transitive dependencies from your Maven and Gradle projects. • View open source licenses and security vulnerability details. • Link the user to the connected Hub instance for additional security detail. The Hub-Eclipse plugin provides inspection of your workspace projects, and gives you an overview of the transitive and direct component dependencies in the project's Maven or Gradle cache. It shows you: • Component • Component version • Component license • Vulnerability count, sorted by severity
Continuous Integration	TeamCity is an open source continuous integration tool that monitors executions of repeated jobs, such as building a software project or cron jobs. TeamCity focuses on building and testing software projects continuously and monitoring execution of externally run jobs. As a Hub and TeamCity user, the Hub-TeamCity plugin enables you to: • Run a component scan in a TeamCity job: ◦ Scan multiple targets within the job workspace. ◦ Define the component scan command line interface (CLI) as a tool. ◦ Create projects and releases in Black Duck Hub through the TeamCity job. ◦ Associate the scanned code with the project/version in the Hub. • After a scan is complete, the results are available on the Hub server. Using the Hub-TeamCity plugin together with Hub Scanner lets you use TeamCity to automatically create Hub projects from your TeamCity projects. Go to the online help for more information on component scanning.

JavaScript

Category	Description
Summary Description	The Hub supports JavaScript discovery with a few notable exceptions. It can scan and identify JavaScript from NPM that has not been minified beyond the original artifact. In most cases, The Hub detects standard downloads of pre-minified libraries.
Discovery Best Practices	For most JavaScript development, Black Duck recommends scanning a file system that includes any custom developed JavaScript and/or any external open source JavaScript packages. However, if Node.js and npm are being used, additional options exist: 1. Using "shrinkwrap" files with npm: these files lock down the respective npm modules that are in use, down to a specific version. The shrinkwrap files can then be examined to determine the list of node.js modules that are in use. Note that this is similar to how Ruby gemfile.lock files are processed. However, using shrinkwrap files in npm is not as standard of a practice as using gemfile.lock files are with Ruby. 2. In the absence of a shrinkwrap file: the scan leverages npm commands to determine a hierarchy of npm models to use. This is basically the equivalent of the npm `ls` command to get a list of installed packages. This does require the packages exist on the system and that npm also be installed on the same system as the scan client. An "npm install" will need to be run before scanning. Therefore, you will be scanning both the actual npm module package artifacts *and* getting the list of dependencies. 3. In the absence of npm being installed: the scan attempts to determine packages by examining the individual package.json files, but determining all the component and the actual versions may be difficult. In most cases, option #2 provides the most comprehensive results, especially depending upon the completeness and accuracy of their shrinkwrap files. Of special interest will be those components only detected by package management and those only detected at the file level. **Javascript Client Side**. Runs directly on a browser (website content). Client side Javascript should be scanning source code that has not been minified. **Javascript Server Side**. Runs on a web server (Node.JS). Server side Javascript (Node.JS) should make use of the npm shrinkwrap file.
Signature Scanning Source	Structure matching as well as signature matching on *.js files.

Category	Description
Code	
Signature Scanning Binaries	SHA1 matching on the following cross language binary types and object code: dll, exe, so, and o.
Signature Scanning Archives	N/A
Signature Scanning Qualifier	Matches are to unmodified and modified collections such as directories, sub-directories, archives and also to virtually unmodified individual .js files.
Build/Package Analysis	Potential future investigation.
Declared Package Inspection	• PM - npm-shrinkwrap.json • NPM - package.json
Primary KB Signature Content Sources	• https://www.npmjs.com/

Linux distributions

Category	Description
Summary Description	Linux distribution (distro) files are matched using standard structure matching.
Discovery Best Practices	Create a directory that contains the Linux distro packages to scan. Black Duck recommends that these packages are in the native formats as received from the distro vendor. They do not need to be unpacked. For example, scan the .rpm files for RedHat distros, .deb files for Debian, and so on.
Signature Scanning Source Code	Structure matching on file system directories and archives.
Signature Scanning Binaries	SHA1 matching on the following cross language binary types and object code: dll, exe, so, and o.
Signature Scanning Archives	Various formats for distros.
Signature Scanning Qualifier	All matches are to collections such as directories, sub-directories, and archives, including unmodified and modified files. Typically packages are: Debian (*.deb, *.udeb), Red Hat Enterprise Linux (RHEL) and Fedora (RPMs), and Alpine (*.apk - tar format).
Build/Package Analysis	N/A
Declared Package Inspection	N/A
Primary KB Signature Content Sources	Debian, RHEL, Fedora, and Alpine

Objective C

Category	Description
Summary Description	Objective C files are matched using standard structure matching.
Discovery Best Practices	Black Duck recommends scanning the build inputs. This includes the source code which is being compiled for the application as well as any third party binaries or libraries that are included with the application. If the third party libraries are not standard binary distributions as provided from common download sites or if those binaries are created internally by compiling the source distributions, Black Duck recommends that the original source for those projects are scanned as well. As structure signature matching is the primary method that is used to find open source, Black Duck recommends scanning the open source in as close to the original structure as it would have been downloaded as possible.
Signature Scanning Source Code	Structure matching on file system directories and archives.
Signature Scanning Binaries	SHA1 matching on the following cross language binary types and object code: .dll, .exe, .so, and .o files.
Signature Scanning Archives	N/A
Signature Scanning Qualifier	• All matches are to collections such as directories, sub-directories. • No match to individual .m files
Build/Package Analysis	Future investigation.
Declared Package Inspection	Future investigation.
Primary KB Signature Content Sources	Cocoapods

PHP

Category	Description
Summary Description	PHP files will be matched using standard structure matching.
Discovery Best Practices	Black Duck recommends scanning the build inputs. This includes the source code which is being compiled for the application as well as any third party binaries or libraries that are included with the application. If the third party libraries are not standard binary distributions as provided from common download sites or if those binaries are created internally by compiling the source distributions, Black Duck recommends that the original source for those projects are scanned as well. As structure signature matching is the primary method that is used to find open source, Black Duck recommends scanning the open source in as close to the original structure as it would have been downloaded as possible.
Signature Scanning Source Code	Structure matching on file system directories and archives.
Signature Scanning Binaries	SHA1 matching on the following cross language binary types and object code: dll, exe, so and o.
Signature Scanning Archives	N/A
Signature Scanning Qualifier	• All matches are to collections such as directories or sub-directories. • No matches to individual .php files,
Build/Package Analysis	Future investigation.
Declared Package Inspection	Future investigation.
Primary KB Signature Content Sources	https://packagist.org/

Python

Category	Description
Summary Description	The Hub scans and identifies Python packages from PyPI and Python and their dependencies as distributed by Anaconda.
Discovery Best Practices	Black Duck recommends scanning the build inputs. This includes the source code which is being compiled for the application as well as any third party binaries or libraries that are included with the application. If the third party libraries are not standard binary distributions as provided from common download sites or if those binaries are created internally by compiling the source distributions, Black Duck recommends that the original source for those projects are scanned as well. As structure signature matching is the primary method that is used to find open source, Black Duck recommends scanning the open source in as close to the original structure as it would have been downloaded as possible.
Signature Scanning Source Code	Directories and sub-directories of .py files and their dependencies.
Signature Scanning Binaries	Matches to unmodified and modified collections of binary code re-used as provided by the open source project.
Signature Scanning Archives	N/A
Signature Scanning Qualifier	• All matches are to collections such as directories, sub-directories, and archives, including unmodified and modified files. • No match to individual Python binaries.
Build/Package Analysis	The Hub-Pip Plugin is available on GitHub. This plugin for Python adds visibility into PIP, a popular package manager for Python packages. It generates a dependency list inside the Hub for your Python projects. Features include: • Capture dependency information from your Setup.py. • Capture dependency information from your requirements.txt (if provided). • Ability to check for policy violations. For additional Python information, refer to: https://docs.python.org
Declared Package Inspection	Future investigation.
Primary KB Signature Content Sources	• https://pypi.python.org/pypi • https://www.continuum.io

Ruby

Category	Description
Summary Description	The Hub scans and identifies directories and sub-directories of Ruby source and modified and unmodified .gem files from rubygems.org. The scan can also analyze gemfile.lock files to get a list of dependencies and populate a BOM, without the presence of the .gem files.
Discovery Best Practices	If you have the Hub version 3.4 and later, the gemfile.lock file (which should exist in the SCM if standard Ruby development processes are followed) is automatically parsed and creates a second Hub scan file that contains the detected dependencies. This scan file will be automatically merged into your BOM. In Hub releases prior to 3.4, scan both your proprietary Ruby source files checked out from SCM and the actual Gem artifacts that are resolved during a bundle process: 1. Check out the code from SCM. 2. Do a bundle install and resolve the dependencies to a directory in your project. You can use the "--deployment" parameter which will copy the dependencies to /vendor/bundle directory in your project or use `--path=<path>` to copy them to a different location. 3. Perform a Hub scan on a directory that contains both the checked out code and the resolved dependencies.
Signature Scanning Source Code	Structure matching on file system directories and archives.
Signature Scanning Binaries	Unpacking and structure matching on unmodified and modified .gem files.
Signature Scanning Archives	Typically .gem files for bundled ruby code.
Signature Scanning Qualifier	All matches are to unmodified and modified collections such as directories, sub-directories, and archives, including .gem packages. There are no matches to individual .rb files.
Build/Package Analysis	Potential future investigation.
Declared Package Inspection	Bundler - Gemfile.lock
Primary KB Signature Content Sources	https://rubygems.org

Scala

Category	Description
Summary Description	Scala support overlaps with Java as Java binaries are typically incorporated in Scala applications. The Hub supports Scala projects from Maven Central, in addition to support for Java components that are frequently combined with Scala projects. Plugins are also available for Maven, Gradle, and SBT which analyze resolved dependencies at build time and improve BOM accuracy. You can find these plugins on GitHub.
Discovery Best Practices	Scala is similar to Java without package management. Black Duck recommends scanning both build inputs (source from SCM) and the derived build outputs (which are all the .jar files produced and used/included during the build).
Signature Scanning Source Code	Structure matching on file system directories and archives.
Signature Scanning Binaries	SHA1 matching on the following cross language binary types and object code: dll, exe, so, and o.
Signature Scanning Archives	Typically, .jar, .war, and .ear files
Signature Scanning Qualifier	All matches are to collections such as directories, sub-directories, and archives, including unmodified and modified .jar files. No match to individual .scala or .class files.
Build/Package Analysis	• Maven/Gradle • Hub-SBT Plugin. Create BOMs, check policies - all from your SBT build The Hub-SBT plugin provides the ability to generate a Black Duck I/O formatted file containing the dependency information gathered from the SBT project. The file is generated in either the specified folder or defaults to the root of the project. This plugin also has the ability to upload the Black Duck I/O file up to the Hub to create a scan file in the Hub. To generate the file and upload the contents to the Hub, the build.sbt file must have a section for this plugin, and execute tasks specific to this plugin. The Hub-SBT plugin runs automatically with your builds, automatically pushing your build into the Hub and generating a BOM. For more information about SBT and plugins, refer to: http://www.scala-sbt.org/0.13/docs/Using-Plugins.html

Category	Description
Declared Package Inspection	Potential future investigation.
Primary KB Signature Content Sources	Maven Central

Swift

Category	Description
Summary Description	Swift are matched using standard structure matching.
Discovery Best Practices	Black Duck recommends scanning the build inputs. This includes the source code which is being compiled for the application as well as any third party binaries or libraries that are included with the application. If the third party libraries are not standard binary distributions as provided from common download sites or if those binaries are created internally by compiling the source distributions, Black Duck recommends that the original source for those projects are scanned as well. As structure signature matching is the primary method that is used to find open source, Black Duck recommends scanning the open source in as close to the original structure as it would have been downloaded as possible.
Signature Scanning Source Code	Structure matching on file system directories and archives.
Signature Scanning Binaries	SHA1 matching on the following cross language binary types and object code: .dll, .exe, .so, and .o files.
Signature Scanning Archives	N/A
Signature Scanning Qualifier	• All matches are to collections such as directories, sub-directories. • No match to individual .swift files
Build/Package Analysis	Future investigation.
Declared Package Inspection	Future investigation.
Primary KB Signature Content Sources	Cocoapods