OpenChain Compliance Automation

Automating the generation of OpenChain compliance artefacts

Alexander Murphy

2023-09-06

OpenChain

International standard (ISO/IEC 5230) lists requirements for a quality open source licence compliance program. The project homepage is here: https://www.openchainproject.org/; and the specificaiton and workgroup manage the development publicly on GitHub here: https://github.com/OpenChain-Project/License-Compliance-Specification.

The standard provides the following definition:

> 2.1 - compliance artifacts a collection of artifacts that represent the output of a compliance program and accompany the supplied software
>
> Note: The collection may include (but is not limited to) one or more of the following: attribution notices, source code, build and install scripts, copy of licenses, copyright notices, modification notifications, written offers, open source component bill of materials, and SPDX documents.

These data are what we will automate in this demonstration (attribution notices and licence texts specifically, other artefacts may be required but this is not an exhaustive exercise, regardless, it is highly unlikely that the steps in this example will map exactly to your own use-case).

Tools

OpenChain is a non-prescriptive standard. To (ab)use computer science terminology, OpenChain is declarative in nature, it tells you *what* to do, but you decide *how* to go about it. This example is a generalised method (rather, a collection of tools and processes) used frequently at Orcro to generate compliance artefacts.

Spelling

Artefacts - British engligh, artifacts, american english. Former used throughout.

Scancode toolkit

Our favorite OSS SCA tool (https://github.com/nexB/scancode-toolkit). Fully featured, well supported, active community. Straightforward to integrate into existing pipelines (run Python script direct from the repo).

Runs comprehensive battery of regex patterns on source code for licence/copyright matches. Huge test suite.

R

Statistical computing programming language. This is the author's personal preference (a dark art) although Python would work, and bash scripts would also work. No libraries are required, just Base R.

Scripts

These are the "glue" that holds togheter the pipeline. These will all be written in R.

Git

The code we review will be located in a Git repository. *this* repo (OpenChain...) contains a submodule to our demonstration project, OpenBLAS (more on this project below).

Overview

We begin by identifying what data we need to automate. This step is usually resource-intensive, so an arbitrary (and vetted) software product will be used for this example.

Then we break-down the full pipeline into "components", and semi-manually run these to illustrate what the tools are doing. I.e., we run through producing the compliance artefacts once step-by-step, and see what happens when we run a "generate compliance artefacts" command in CI.

The output of the process (the compliance artefacts) will be shown. These will be in the form of plain text files, which can be compressed if required.

Setting expectations

Implementing the process on a CI system is not covered.

This only covers generation of compliance artefacts for the application itself. It is likely that additional FOSS is used in any shipped product (Docker container, GPU drivers, etc.) but this process is for the application layer only.

We won't consider legal requirements, such as how the jurisdiction your code is shipped to may affect your obligations.

We won't consider engineer considerations, such as caching SCA results.

The sample project

OpenBLAS

BLAS (basic linear algebra subprograms) is a widely used library for matrix and vector calculations. It's open source, the reference implementation (from https://www.netlib.org/blas/) isn't "licensed" per se, rather it has some unusual wording about being free to use but to add notices if anything is modified and to provide acknowledgement for the authors.

OpenBLAS has the BSD-3-Clause out-licence, and the full source (which we look at here) is available on GitHub (https://github.com/xianyi/OpenBLAS).

Identifying the requried artefacts

Questions to ask

What is *distributed*?

Are there any *dependencies*?

How will the code be shipped?

Are there any *snippets* present?

In this case, all of the source code is provided in this monorepo. We can run our tools here without requiring any additional downloads.

Recall that build tools won't end up in the distributed binary, therefore you have no need to concern yourselves with licences for them (typically, but never seen a case where this is not the case).

Licences are BSD-3-Clause, therefore our obligations are simply: Provide copyright notices (all of them!) and the licence text. * another obligation is to not use the name of the project in marketing, but this does not impact the compliance artefacts r.e. openchain

We also see that it doesn't matter whether the code is distributed as source or in binary form. The artefacts are required* note that if distributed as source code, then the compliance materials *will be provided automatically as they are contained within the source itself!*

For this we assume the binary is distributed, and providing source code is unsuitable for some reason.

Note: providing the full source code for components licensed permissively is not *required* but it may be one way to comply! Although, if you have more than one component with different obligations then things can get messy quickly, hence, automation...

Initial SCA

It is prudent to manually run a SCA before implementing any automation. Like shopping for bug-spray, it is important to know *what will work* before spending any money (or developer time, resources, etc.) on the thing. It would be disastrous to implement compliance automation and a year down-the-line realise that it wasn't identifying issues appropriately (or even worse, it *was* identifying them but you didn't notice)!

Scancode can be run from the command-line. So a developer, who is working inside the development repo, can simply use:

```
pwd
/home/demo/OpenChain-Artefact-Automation/OpenBLAS/
scancode -clp -n 18 --csv ScanOut.csv .
```

which will run scancode on the source repo. This assume that scancode has been installed (added to path) et cetera (see Appendix A for simple scancode installation instructions).

Appendix A: Scancode installation

`cd` to the installation directory, then

```
git clone --depth=1 https://github.com/nexB/scancode-toolkit.git
cd ./scancode
./scancode --help
```

Running `./scancode --help` will setup scancode for first use. There are other methods in the documentation (https://scancode-toolkit.readthedocs.io/en/stable/) but the above is, clearly, very simple. The above can be copied into a `.sh` script, Docker `RUN` directive, et cetera, as required.