Simple License Policy from Lawyer to Computer

FSFE Legal Workshop 12 April 2025

Martin von Willebrand and Vladimir Slavov

Contents

Introductions

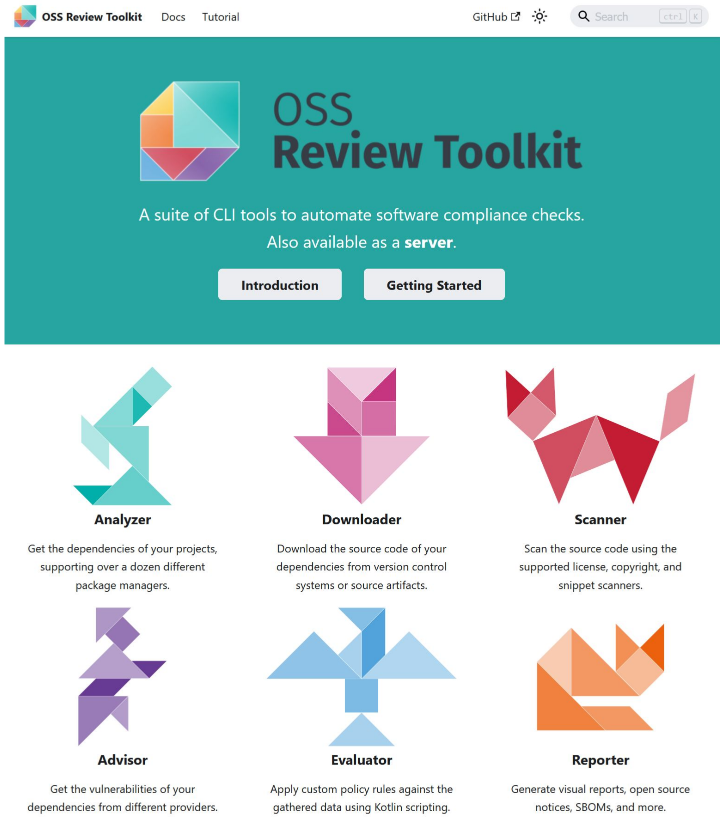
Martin von Willebrand

- Attorney, Chair, HH Partners, Attorneys-at-law
- Co-Founder and Chair, Double Open
- Chair, Validos

Vladimir Slavov

- OSPO @ Bosch Digital
- fmr. Open Source Officer @ Bosch.IO
- Committer @ Eclipse Apoapsis
- Lawyer & Programmer
- Certified AWS Solution Architect Associate
- Certified AWS Cloud Practitioner

Example automation implementation



OSS Review Toolkit Docs Tutorial GitHub Search ctrl K

OSS Review Toolkit

A suite of CLI tools to automate software compliance checks.
Also available as a **server**.

Introduction | Getting Started

Analyzer

Get the dependencies of your projects, supporting over a dozen different package managers.

Downloader

Download the source code of your dependencies from version control systems or source artifacts.

Scanner

Scan the source code using the supported license, copyright, and snippet scanners.

Advisor

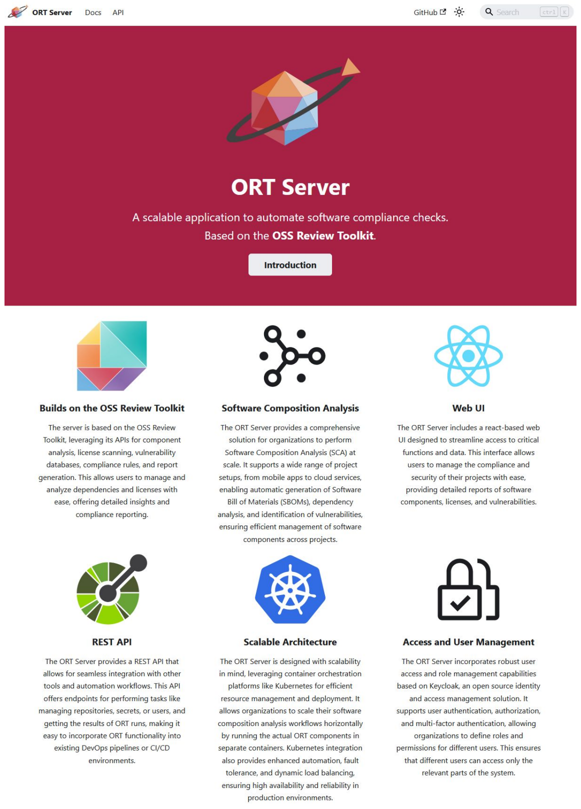
Get the vulnerabilities of your dependencies from different providers.

Evaluator

Apply custom policy rules against the gathered data using Kotlin scripting.

Reporter

Generate visual reports, open source notices, SBOMs, and more.



ORT Server Docs API GitHub Search ctrl K

ORT Server

A scalable application to automate software compliance checks.
Based on the **OSS Review Toolkit**.

Introduction

Builds on the OSS Review Toolkit

The server is based on the OSS Review Toolkit, leveraging its APIs for component analysis, license scanning, vulnerability databases, compliance rules, and report generation. This allows users to manage and analyze dependencies and licenses with ease, offering detailed insights and compliance reporting.

Software Composition Analysis

The ORT Server provides a comprehensive solution for organizations to perform Software Composition Analysis (SCA) at scale. It supports a wide range of project setups, from mobile apps to cloud services, enabling automatic generation of Software Bill of Materials (SBOMs), dependency analysis, and identification of vulnerabilities, ensuring efficient management of software components across projects.

Web UI

The ORT Server includes a react-based web UI designed to streamline access to critical functions and data. This interface allows users to manage the compliance and security of their projects with ease, providing detailed reports of software components, licenses, and vulnerabilities.

REST API

The ORT Server provides a REST API that allows for seamless integration with other tools and automation workflows. This API offers endpoints for performing tasks like managing repositories, secrets, or users, and getting the results of ORT runs, making it easy to incorporate ORT functionality into existing DevOps pipelines or CI/CD environments.

Scalable Architecture

The ORT Server is designed with scalability in mind, leveraging container orchestration platforms like Kubernetes for efficient resource management and deployment. It allows organizations to scale their software composition analysis workflows horizontally by running the actual ORT components in separate containers. Kubernetes integration also provides enhanced automation, fault tolerance, and dynamic load balancing, ensuring high availability and reliability in production environments.

Access and User Management

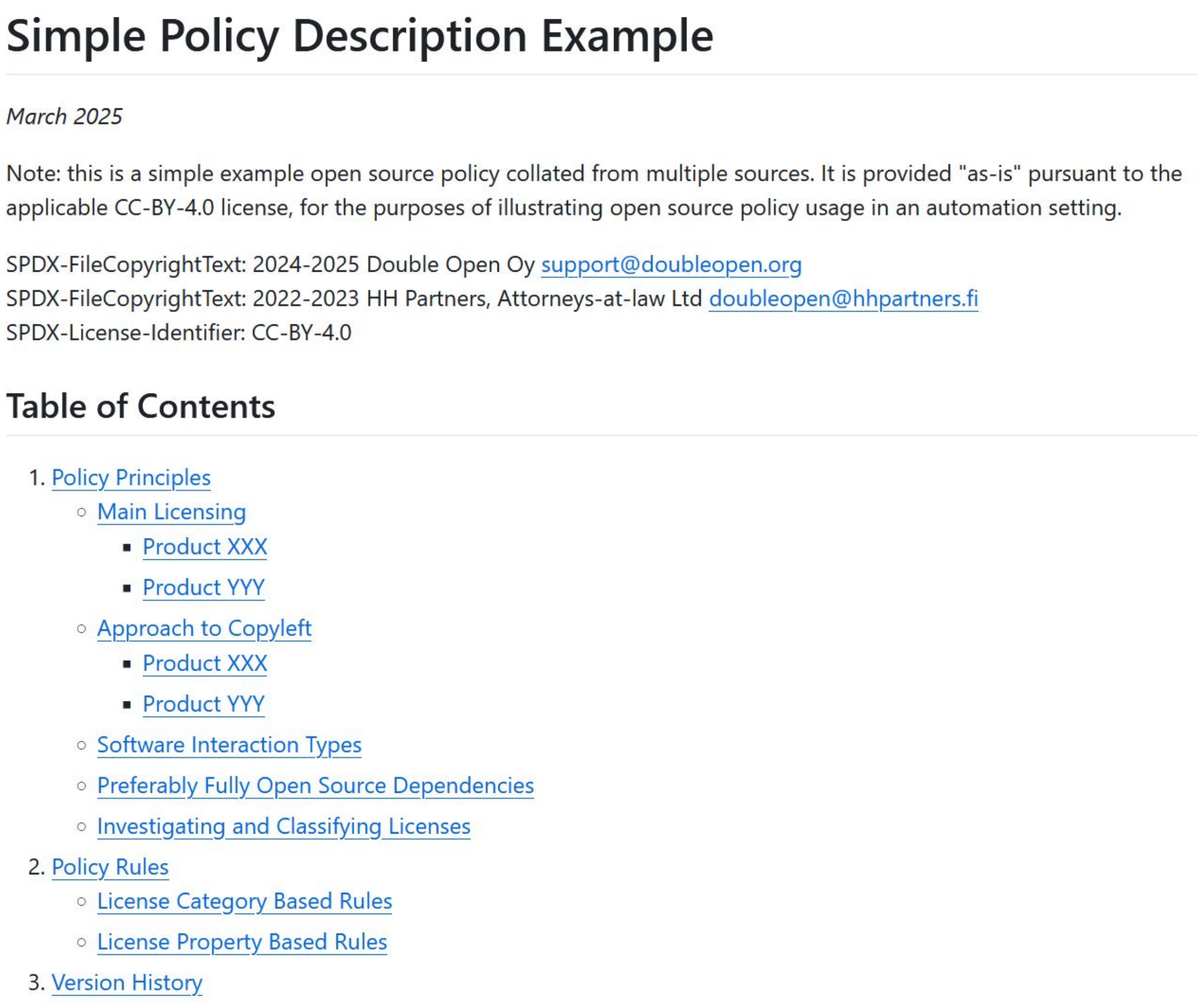
The ORT Server incorporates robust user access and role management capabilities based on Keycloak, an open source identity and access management solution. It supports user authentication, authorization, and multi-factor authentication, allowing organizations to define roles and permissions for different users. This ensures that different users can access only the relevant parts of the system.

doubleopen-project / dos

Code | Issues 8 | Pull requests 1

dos Public

Main licenses: Apache-2.0 AND MIT



Simple Policy Description Example

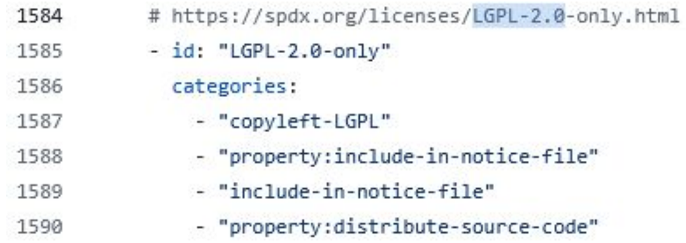
March 2025

Note: this is a simple example open source policy collated from multiple sources. It is provided "as-is" pursuant to the applicable CC-BY-4.0 license, for the purposes of illustrating open source policy usage in an automation setting.

Table of Contents

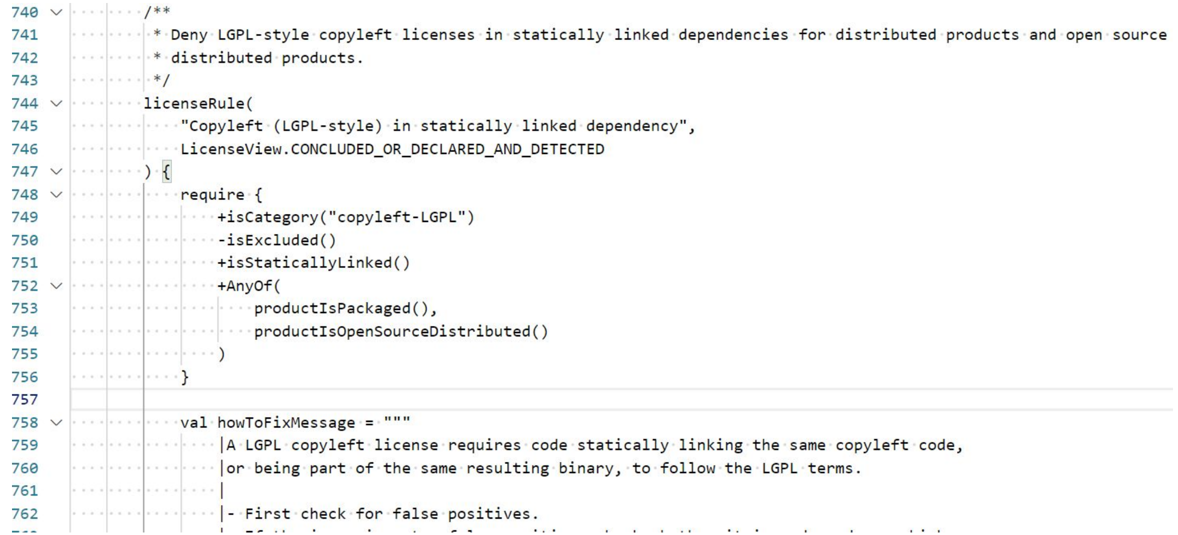
Main licenses:
CC-BY-4.0

License classification



```
# https://spdx.org/licenses/LGPL-2.0-only.html
- id: "LGPL-2.0-only"
  categories:
    - "copyleft-LGPL"
    - "property:include-in-notice-file"
    - "include-in-notice-file"
    - "property:distribute-source-code"
```

Evaluator rules



```
    /**
     * Deny LGPL-style copyleft licenses in statically linked dependencies for distributed products and open source
     * distributed products.
     */
    licenseRule(
        "Copyleft (LGPL-style) in statically linked dependency",
        LicenseView.CONCLUDED_OR_DECLARED_AND_DETECTED
    ) {
        require {
            +isCategory("copyleft-LGPL")
            -isExcluded()
            +isStaticallyLinked()
            +AnyOf(
                productIsPackaged(),
                productIsOpenSourceDistributed()
            )
        }

        val howToFixMessage = """
            |A LGPL copyleft license requires code statically linking the same copyleft code,
            |or being part of the same resulting binary, to follow the LGPL terms.
            |
            |- First check for false positives.
```

Main licenses:
CC0 AND
CC-BY-4.0 AND
Apache 2.0

License policy examples

Repo for examples: https://github.com/doubleopen-project/license-policy-demo

Human readable policy: https://github.com/doubleopen-project/license-policy-demo/blob/main/policy_example.md

Kotlin script policy: https://github.com/doubleopen-project/license-policy-demo/blob/main/example.rules.kts

License classification: https://github.com/doubleopen-project/policy-configuration/blob/main/license-classifications.yml

Main licenses:
CC0 AND
CC-BY-4.0 AND
Apache 2.0

Introduction to Compliance Logic / scanner false positive

Compliance result	PASS
Policy violation resolution	No
Evaluation rules	evaluator.rules.kts
Product label	Distributed product
License categorization	MIT: - "permissive" - "property:include-in-notice-file"
Human curations	MIT
Scanner/DB license data	GPL-1.0-or-later
License metadata by project	MIT
Artefact	pkg:gem/algoliasearch@1.27.5

Introduction to Compliance Logic / scanner positive

Compliance result	Policy violation
Policy violation resolution	No
Evaluation rules	evaluator.rules.kts
Product label	Distributed product
License categorization	GPL-2.0-only: - "copyleft-strong" - "property:include-in-notice-file"
Human curations	GPL-2.0-only
Scanner/DB license data	GPL-2.0-only
License metadata by project	MIT
Artefact	pkg:maven/com.wix/detox@20.34.0 detox/src/.../android/emulator/EmulatorVersionResolver.test.js

Introduction to Compliance Logic / scanner positive

Compliance result	Policy violation
Policy violation resolution	Yes, comments “LICENSE ACQUIRED” “Good architecture.”
Evaluation rules	evaluator.rules.kts
Product label	Distributed product
License categorization	GPL-2.0-only: - "copyleft-strong" - "property:include-in-notice-file"
Human curations	GPL-2.0-only
Scanner/DB license data	GPL-2.0-only
License metadata by project	MIT
Artefact	pkg:maven/com.wix/detox@20.34.0 detox/src/.../android/emulator/EmulatorVersionResolver.test.js

Introduction to Compliance Logic / SaaS label

Compliance result	PASS
Policy violation resolution	No
Evaluation rules	evaluator.rules.kts
Product label	SaaS
License categorization	GPL-2.0-only: - "copyleft-strong" - "property:include-in-notice-file"
Human curations	GPL-2.0-only
Scanner/DB license data	GPL-2.0-only
License metadata by project	MIT
Artefact	pkg:maven/com.wix/detox@20.34.0 detox/src/.../android/emulator/EmulatorVersionResolver.test.js

Automation Pain Points

- Super-configurability
- The Lawyer’s Automation Dilemma
- Defining Use Cases / Business Contexts
- Difficult cases for a mass process
- Fulfilling more exotic obligations

Super-configurability

- Everything in ORT is Configuration-as-Code:
  - licenses and their properties
  - policy rules
  - data curations
  - etc.
- Steep initial adoption curve.
- Majority of effort concentrated in the beginning.
- Upsides:
  - easy reproducibility
  - clear audit trail

The Lawyer's Automation Dilemma

- As lawyers, we may be reluctant to relinquish judgement to a computer in cases with potential legal relevance.
- With increasing scale, there comes a point where performing only manual assessments is no longer possible.
- When that point is reached, it may be *less risky* to automate than not to automate. Automation becomes not only an option but a necessity.
- Question: *How sophisticated* should the automation be? Depends on:
  - The resources that can be dedicated to it.
  - The risk tolerance of the organization.

Defining Use Cases / Business Contexts

- The use of Use Cases / Business Contexts enables a more fine-grained rule configuration.
- The same component may create legal issues in one case but not in another (e.g., a GPL-3.0 licensed dependency in a proprietary mobile app vs. a server backend).

Difficult Cases for a Mass Process

- A significant challenge is automating **architecture-related compliance checks** (e.g. to determine extent of copyleft effect).
  - The ORT way of automatically determining the architecture:
    - File changes: ORT detects dependencies on the package level. Analyzer-result has information if a package is identical to upstream available package.
    - Linking: ORT Analyzer queries the build-system, and stores information for each package (or node) whether a particular package is *linked statically* to the root application.
    - User input: Architecture can be explicitly specified using a *project.spdx.yml* file. Manual maintenance of the description is required.
- Resolving policy violations by putting your own code under an OSS license → typically separate/more difficult process.

Fulfilling Exotic Obligations

- ORT can provide you with:
    - a Disclosure Document
    - a Source Code Bundle
    - SBOMs (SPDX and CycloneDX)
    - various reports
- These can be used to fulfill the majority of OSS-related obligations.
- If an OSS license requires you to buy the copyright owner a beer though, you are out of luck. ☺

Thank you!