TL;DR

Modern technologies that form the backbone of major technological innovations all have roots in open source — from microservices to agile CI/CD, from digital transformation to artificial intelligence (and we could go on). Open source is now a key part of any software strategy, with 77% of Enterprises including open source in commercial products,* and it's not hard to understand why. From lowering the total cost of ownership (and decreasing time to market) to improving development practices and product quality, open source comes with clear benefits to your business.

To ensure your company can reap the benefits from a rapid adoption of open source technologies, you need to have a strategy to manage your open source consumption responsibility. Understanding where to get started, however — especially when you're operating at scale — can be daunting. FOSSA has partnered with leaders in open source from foundations like OpenChain and Open Source Program Office experts from companies like Uber, Verizon Media, Ford, and TDAmeritrade to assemble and share these best practices in developing open source compliance programs.

FOSSA

Creating an Open Source Compliance Program: Auditing your Company's use of Open Source

Phase 1: Create Open Source Compliance Policies

Phase 1 Checklist

Step 1: Understand Your Company's Software Profile

- Create an inventory of company products
- Create product profiles
- Create priority/risk levels for each product

Step 2: Create License Policies

- Create a general license policy for each product type
- [Optional] Create a policy for integrated software from external vendors
- [Optional] Create policy for contribution & publishing
- [Optional] Create Inbound Request Process

Step 3: Create Educational Materials

- Create resources to educate software teams
- Create training for relevant teams

Step 1: Understand Your Company's Software Profile

Open Source Compliance Program Recommendations

1. Prioritize and Iterate your rollout. Trying to implement an overarching compliance program across many teams all at once can be overwhelming and delivers underwhelming results. We've seen that enterprises have the most success when first prioritizing company products, then strategically rolling out compliance initiatives team by team or product by product. This allows for iteration and improvement to reduce any confusion, update any tool configurations, and share internal case studies as your compliance program grows.
2. Automate where possible. Many companies do start managing open source components leveraging open source request forms and spreadsheets. This process decreases accuracy and coverage of any risk assessments or Bill of Materials, and increases the overall cost of a compliance program to a company by inefficiently allocating resources.
3. Process is Key. While many parts of this process can and should be automated, there will always be people involved (this may include selecting new open source projects to implement, refining dependency identification, enriching dependency metadata, reviewing flagged components, or generating reports). Ensure you have a process for each of these stages in your compliance program and opt for tools and workflows that enable team members at each stage.

1. Create an Inventory of Company Products

Generate a list of all products that includes ANY software. ***Ensure you consider the following:***

- Free or paid products that are downloadable or installable on a customer's or partner's machine
- Mobile applications
- SaaS applications
- Web-hosted properties
- SDKs, plugins, tools
- Software embedded in hardware devices
- Operating systems

2. Create Product Profiles

A product profile is a form of knowledge management about products that act as a source of truth for those involved in open source compliance over time. This information gives information about the tools and third-party license policies that should be applied to this product. ***For the simplest version of your Product Profile Inventory we recommend including at least the following fields:***

- Product/project name (linked to code repository)
- Description
- Internally vs. externally facing product (and distribution method)
- Key contact info for software/product owner
- Priority score
- Compliance rollout phase
- Notes

Before you Get Started

Determine Open Source Compliance Program Owners. We recommend assembling a team comprised of:

- Program Owner to own compliance rollout, process, and communication
- Legal Representation to help create policy and interpret licenses
- IT/Engineering to implement any tools
- Engineering leaders

Depending on your company requirements the following information included in the project profile might include:

- Product Metadata
  - Detail about what programming languages are used
  - Brief explanation about what the product does
  - Explanation of how it's accessible/distributed to others
  - Detail about relationships with other products (i.e. this is an on prem component for your SaaS product)
  - How the product is distributed (i.e. container, virtualized application, embedded in hardware, downloaded by customer)
  - If the product is OEMed or might be in the future
- Key Contact Metadata: list the main contacts for
  - Engineering
  - Product Management
  - Release Management
  - Legal disclosures
- Release Metadata:
  - Release schedule
  - Next release date
- Compliance Metadata:
  - Priority score
  - License policy
  - Compliance rollout phase
  - Links to key reports (BOM, attribution)

3. Create Prioritization Score

This portion is entirely dependent on your company. ***A couple of example factors (ranked by importance) to include when prioritizing your products include:***

- Highest distribution (remember to include free products)
- Distribution method (e.g downloaded by customer, SaaS, included in hardware)
- Highest business impact (connected to multiple other products/ systems)
- Release timing/cadence
- Highest percentage of third party code

Step 2: Create License Policies

1. Create a general license policy for each product type

Generally speaking, different product types have different needs. In some products you might have a policy against including any copy-left licenses like GPL while in others your policy might be to publish any projects that include GPL. ***To create a license policy for a given product:***

- Identify “Green” licenses that are always permitted for use in a given product (this often includes permissive licenses like MIT and Apache 2.0)
- Identify “Red” licenses that are never permitted for use in a given product
- Identify “Yellow” licenses with Specific Policies

(e.g. GPL is OK if used as a stand alone executable running in a separate address space).
NOTE: Most Yellow Licenses do not need to be identified as these are generally handled on a case by case basis. ***Examples of conditional Yellow license policies include the following situations:***
- Licenses you can use in some areas of product development
- Licenses you can include depending on how the open source component is linked
- Licenses that require additional obligations

- Create guiding principles for when to approve Yellow Licences

For more information on creating an open source license policy here is a great talk by Kate Downing, a member of the Linux Foundation, at a FINOS working group.

For more information on open source licenses and their obligations a few good references are:

- TLDRLegal: open source licenses in plain English
- Open Source Initiatives: licenses approved by OSI
- The Legal Side of Open Source: GitHub curated resource
- SPDX License List

If you choose to evaluate 3rd party tools to help manage your open source compliance

we recommend evaluating software based on compatibility with existing software development practices, policy priorities, ease of integration, and the correct balance of automation and manual review for your organization.

Example questions you might ask include:

- Can you create a policy inside the tool?
- Can you create different policies for different products?
- Does the platform “auto-approve” green licenses?
- Does the platform have the option to block builds containing red licenses?

2. [Optional] Create a license policy for integrated software from external vendors

When you OEM software from a third-party and incorporate it into your codebase and/or products, you become a distributor of that software and need to ensure that it complies with your open source licensing policies.

3.[Optional] Create Process for contribution & publishing

- Contributions should consider the business benefits such as building a community, code contributions, adoption of technology important to the business.
- Contributions should consider the supportability and maintainability benefits of providing fixes and enhancements to upstream projects

- Contributions should consider the licensing and contributor agreements for existing open source projects
- Licensing for company sponsored open source projects should be carefully considered based on the desire for adoption and the desire to ensure modifications are contributed back

4. [Optional] Create Request Process
If you decide to implement an inbound request policy collect the following information so you can credibly determine whether or not the open source component can be used:

- Dependency name and version
- Declared license (if any)
- Link to dependency source code
- Description of how the dependency will be used
- Application/product the dependency will be integrated into

We recommend scanning each inbound request to identify and deep dependencies and their licenses.

Step 3: Create Educational Materials

1. Create resources to educate software teams
We recommend creating an internal document repository to point any questions, and to serve as a continual resource post-training. ***In the wiki we recommend including the following information:***

- Overview of licenses and their categories (Green,

Yellow, Red)

- Key contacts to direct questions to:
  - Legal contact
  - Open source technical contact
  - Open source tooling owners
- Key resources and tooling information

2. Create training for relevant teams

Establish understanding across each team. We recommend focusing your training on understanding why compliance is important, understanding general policy, and understanding the resources available.

Phase 2:
Initiate Compliance Program Roll Out

Phase 2 Checklist

Step 1: Identify High Priority Product(s) & Assemble Team

- Identify highest priority products to start with
- Identify key stakeholders from relevant team(s)
- Identify correct license policy
- Complete team compliance training

Step 2: Dependency & License identification

- Identify third-party open source components for individual products
- Identify third-party open source component metadata

Step 3: Issue Identification & Remediation

- Leveraging your policy–identify components with flagged licenses
- Review component metadata to determine next steps
- If remediation is required, work with relevant team to address the specific issues (e.g. replace the component, publish proprietary code, and/or add required notices).

Step 4: Fulfilling Open Source Obligations and Generating Attribution Files

- Create a Bill of Materials by compiling all third party components and their licenses (SPDX recommend format)
- Review included licenses for obligations to determine next steps

Step 5: Finalize Process & Product Compliance

- Ensure you have correct attribution reports, source code is made available when required, there are no outstanding remediation issues, and all obligations have been fulfilled
- Ensure system is in place to continue to implement this process for new commits to the code base with clear owners
- Reflect on improvements for subsequent products

Step 1: Identify High Priority Product(s) & Assemble Team

- Identify highest priority products to begin rollout
- Identify key stakeholders from relevant team(s). This is generally the engineering or software development leader of each team
- Identify correct license policy
- Complete team compliance training

Step 2: Dependency & License identification

1. Identify Third-Party Open Source Components for Individual Products

2. Identify Third-Party Open Source Component Metadata

We recommend automating this part of the process as much as possible. There are both enterprise-grade and open source tools that automate dependency identification to a varying extent. ***You need to gather the following information:***

- Dependency name, version

- Copyright information
- Declared license
- Embedded licenses
- Deep dependencies (also referred to as transitive dependencies)
- Copyright information
- Declared license
- Embedded licenses
- File where found
- Inclusion path (dependency tree)

When evaluating a solution to help automate the identification of dependencies you should evaluate the following:

- The maturity of the solution
- The support available for the solution
- The sustainability of the solution
- The ease of integration with your existing processes
- The languages/technology supported (i.e. does the solution support your companies build tools/ development processes)

Step 3: Issue Identification & Remediation

Depending on your process and policies this can take quite a bit of effort. We recommend evaluating 3rd Party software to aide in the identification and

remediation of issues. ***Common evaluation criteria include:***

- Auto approval for all green licenses
- Issue prioritization
- Issue context including information like:
  - Code that triggered the flag
  - Access to the dependency code
  - Information about how the component was pulled into your codebase
  - File path to the dependency
- Integrations with engineering task management tools like JIRA
- Alerting via email, Slack, etc.
- Manual resolution
  - Ignore – with reason
  - Dependency enrichment
  - License enrichment

1. Leveraging your policy – identify components with flagged licenses

- Identify all components, both direct and deep dependencies, with “Yellow” and “Red” licenses

2. Review component metadata to determine Next Steps

This is likely to be the part in the process that requires the most hands-on attention. ***When determining whether or not an issue requires attention consider information like:***

- The source of information

- What is the linkage?
- How is the component used?
- Is it dual-licensed?
- Scanned license/copyright data from source code

3. If remediation is required, work with relevant team to resolve the issue

Potential remediation can include:

- Publishing code based on or linked to the open source software. This may include proprietary code.
- Removing or replacing the flagged component and re-architecting your software
- Fulfilling additional obligations like stating changes or building an attribution notices

Step 4: Fulfilling Open Source Obligations and Generating Attribution Files

1. Create a Bill of Materials (SPDX recommend format)

Generating a bill of materials or attribution notice can be automated by several tools or compiled by hand. ***When generating a Bill of Materials you generally include the following fields:***

- Package name
- Creator
- License
- And others such as listed here: https://spdx.org/sites/cpstandard/files/pages/files/spdx_

onepager.pdf

2. Review included licenses for obligations to determine next steps

Depending on your obligations ensure you have:

- Published/provided any source code for copy-left licenses
- Published your attribution notice, stating any changes, source, etc. as deemed necessary by the license
- Repeated the above process if any software was modified during the process

Step 5: Finalize Process & Product Compliance

1. Ensure you have correct attribution reports, there are no outstanding remediation issues, and all obligations have been fulfilled

2. Ensure there is a system in place to continue to implement this process for new commits to the code base with clear owners

Determine your frequency. Several tools can be integrated into the CI/CD pipeline to run continuously in the background with every code comment. Best practice is to ensure you are at least compliant with every deploy and/or product release.

2. Reflect on improvements for subsequent products

Phase 3:
Iterate & Repeat with Subsequent Products

Phase 3 Checklist

Step 1: Track Program Rollout

- We recommend monitoring coverage as a percent of product

Step 2: [Optional] Key Actionable & Success Metrics

- Issues by team or by product: determine which teams need more hands-on assistance and training
- Time-to-issue resolution: monitor the progress and efficiency of your systems and processes

About FOSSA

FOSSA can help to achieve all of these best practices. By providing automated, real-time licensing and vulnerability management for open source code no matter where it exists within your software stack, FOSSA helps organizations minimize the risk and maximize the benefit of open source. Request a demo to learn more, or import FOSSA from GitHub to start analyzing your open source dependencies today.

fossa.com