



License-Compliant Distribution of Open Source Code

Dirk Riehle , Friedrich-Alexander-Universität Erlangen-Nürnberg

This column previously discussed how to prepare your project or product by complying with key obligations like attribution and copyleft.

This column now explains how to package up your project or product and how to distribute it in a license-compliant way.

As you distribute your program to recipients, you have to provide them with the proper legal notices for any open source code included in your distribution.

Your distribution may be binary only, or corresponding source code only, or binary with corresponding source code. If you distribute corresponding source code, most lawyers will argue that you have already fulfilled the obligation to provide proper legal notices because the notices

are included in the source code. However, most commercial software developers prefer to distribute binary code only, keeping their source code to themselves.

As discussed previously, the first step then is to collect all legal text snippets (copyright notices, license texts, other notices) from all included open source code, usually during software composition analysis. The second step is to compile these text snippets into a single file, the legal notices. A single file is not required by the licenses; however, it is a best practice corresponding with the in-

tent of the licenses, which is to inform recipients of the distribution about the open source code they are receiving.

Figure 1 shows how my Android phone makes its open source legal notices available to users of the phone.

As always, the best possible advice for complying with licenses is to work in line with their intentions and not against them, even if the written word may allow for multiple interpretations. The Android legal notices illustrate legal notices done well:

1. First, there are no barriers to access. If you receive an Android device, you are typically able to access



FROM THE EDITOR

Complying with open source licenses not only requires the creation of proper legal notices and, if applicable, corresponding source code, it also requires proper packaging and distribution. Following up on a previous instance of the column, this article discusses how to put the puzzle pieces together, correctly, and distribute it to recipients. Happy hacking, and be healthy and happy.—Dirk Riehle

it and find your way to the legal notices.

- 2. Second, the legal notices are easy to find. While “easy” may be relative, the Android legal notices can be found in a place that makes sense.
3. Third, they are logically structured. The Android legal notices start with a table of contents

that lets you jump to the information for each component.

- 4. Fourth, they have the user in mind. The legal notices show not only the required, but additional information as well (here, where the code is used).

Hiding the legal notices behind a login or obstructing their access in any other way violates the intent, though not necessarily the legal text, of the licenses.

While creating a single legal notices file is a widely accepted best practice, how to bring this file in front of users is a context-dependent decision. This decision strongly depends on what you are delivering to recipients. Your options are 1) to provide printouts, 2) to provide DVDs with the legal notices file, or 3) to provide software access in the delivered binary (like in the Android example).

The closer the legal notices are to the (binary) code, the better.

You should not just provide a link to a file on the web; the legal notices themselves must be part of the delivery.

Copies on a DVD are the primary option if you are providing products with limited or no access to an electronic version of the legal notices. Examples of such products range from small inexpensive integrated circuits (and their firmware) all the way to large objects like cars. If you deliver in bulk, you can deliver the legal notices once with every shipment.

In a running program, access to legal notices should be easy and feel natural to a user. If the program is primarily used through a command line interface, there should be a command line option to receive the legal notices. If the program is primarily used through a graphical user interface, there should be a menu entry that provides the legal notices. The menu entry for the open source legal notices should be close or next to any other legal notices.

THE LICENSE-COMPLIANT DELIVERY WORKFLOW

Shipping a product that includes open source code to customers requires

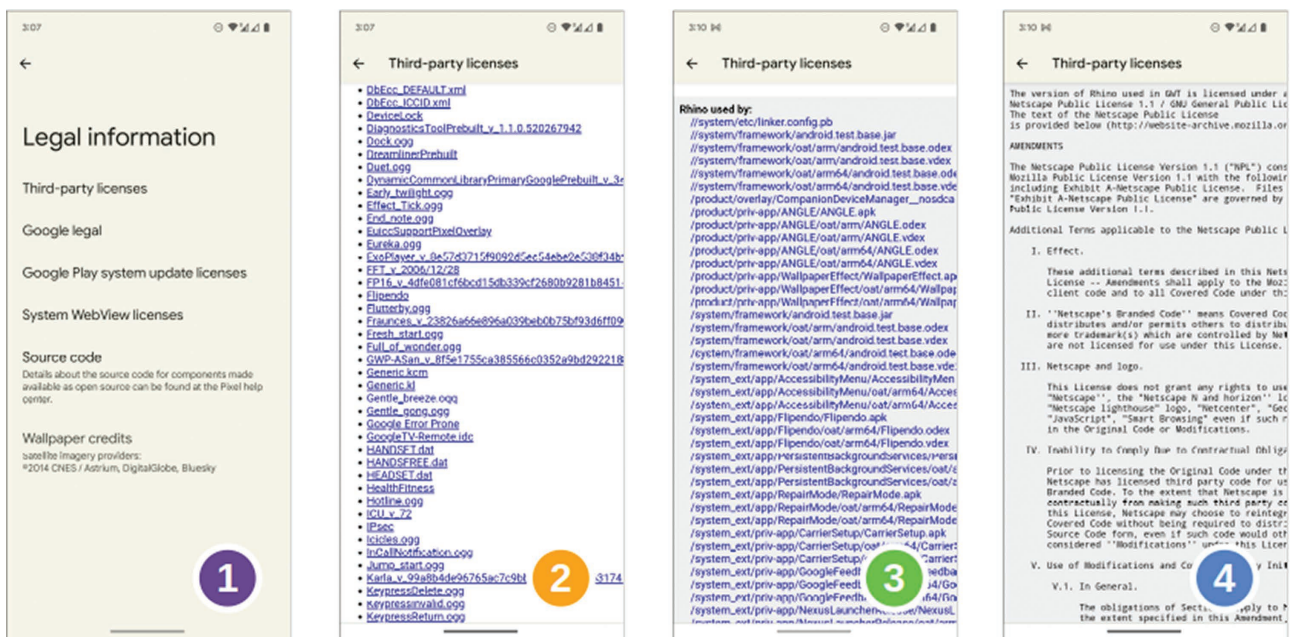


FIGURE 1. Android phone open source legal notices.

compliance with the licenses of the included code. For most products and in most companies, distributing a binary with open source code is a complex workflow that is best managed by an open source program office (OSPO) with the help of appropriate tooling like a SCA Tool.^a To recap: An OSPO is an organizational unit inside a company tasked with bringing order, best practices, and efficiency to using open source software as well as contributing to and leading open source projects.

Hiding the legal notices behind a login or obstructing their access in any other way violates the intent, though not necessarily the legal text, of the licenses.

The license-compliant distribution workflow of projects or products that contain open source code starts long before the creation and provision of legal notices or corresponding source code to customers. It also doesn't end there. The base workflow contains at least these activities:

1. review and approval of incoming open source components (see “inbound licensing”)
2. building of the project or product binaries including fully tracked and archived source code
3. review and approval of outgoing software with open source code (see “outbound licensing”)
4. creation of legal notices or corresponding source code (as needed)
5. license-compliant distribution of the project or product.

Base activity 1, the review process, includes the software composition analysis of the incoming open source component to determine the component's software bill of materials (SBOM). The approval of the component

includes reviewing the licenses of the materials in the component's SBOM.

Base activity 2, building the project or product, includes the provision and usage of build tools and corresponding configurations of these tools. Pre-commit hooks may want to test that a commit has not been copied from Stackoverflow, component repositories may be used to provide vetted versions of approved components, and/or scripts and version control systems may be used to set aside correct

corresponding source code for the binary build being created.

Base activity 3, review and approval of outgoing projects or products, includes the review of the to-be-distributed code beyond the original approval. Outbound licensing knows more than inbound licensing: The software architecture may have changed, creating an unwanted copyleft effect where previously there was none, an unwanted patent grant may follow from compliance with an open source license, and/or the recipient of the distribution may reside in an embargoed country so export control rules need to be observed.

Base activity 4 then proceeds as outlined in a previous section, ideally by relying on a properly compiled SBOM to create the legal notices. Ideally, this activity can be merged into base activity 2, building the binary in the first place. However, tools often aren't that well integrated, making the creation of legal notices a separate activity.

Base activity 5 then ensures that the legal notices will be incorporated into the distributed binary in an appropriate way, also as discussed in a previous section.

The operations of this base workflow are typically performed by the

engineering organizational units. In parallel, the OSPO needs to operate a meta workflow that accompanies and guides the base workflow. The meta workflow contains at least these activities:

1. provision and operation of compliance tools to support the engineering organizational units
2. provision of configurations of compliance tools to establish and maintain corporate standards
3. provision of a central public point of contact for open source related questions.

Meta activity 1, the provision of compliance tools, includes the licensing and operations of SBOM management, open source governance, and license compliance tools. Engineering organizational units will typically run their own build processes but will benefit from pulling in compliance tooling from a centrally managed place.

Meta activity 2, the provision of configurations, includes the configuration of said compliance tools. Such configuration can be nontrivial, including custom development work. The OSPO may have to set up a standardized license interpretation for the company's products and encode it in the tools. This way, the OSPO makes sure that different organizational units still treat open source licenses the same way. OSPOs that rely on a tool vendor's off-the-shelf configuration do so at their own risk.

Meta activity 3, the provision of a public point of contact for open source related questions is a best practice in which all projects and products of the company that contain open source code provide an address of who to talk to with questions about open source code in a binary. Usually, it is just an email inbox, but it needs to be monitored and inquiries need to be answered.

There are more activities you may have to consider. Typically, the larger

^aSee <https://scatool.com>.

the company and its product portfolio, the more activities there are.

LEGAL THREATS AND RESOLUTIONS

Obviously, you should follow the licenses of open source code included in your projects and products when distributing this code to third parties, for example, when shipping your product to customers. Beyond the moral and legal argument, however, you may wonder: How important is it? After all, the software will do its job just fine, irrespective of whether it comes with correct legal notices or not.

Like most business decisions, this is an exercise in risk management. How much time and effort you spend on license compliance depends on your risk of getting sued for any license violations. Thus, a company has to assess its risk profile. Variables of the risk profile are market (consumer versus enterprise), type of software (embedded versus application), type of distribution (on-premise versus cloud) and others.

There are two main sources of potential lawsuits:

1. *Copyright and patent trolls*: Trolls are individuals or companies that sue you to extract as much money as possible; it is their business model.
2. *Copyleft enforcers*: Enforcers will sue you specifically for copyleft violation; their origins are in the free software movement.

Copyright trolls use the copyright they hold in an open source software to extract money from anyone who violates their rights, and patent trolls use patents they own to extract money from anyone who uses open source software that utilizes these patents, knowingly or not. In general, both copyright and patent trolls search the web for potential violators, and, if they find someone, start their work.

Copyright trolls look for missing or incomplete legal notices. If they find a software that is distributed to third

parties in binary form and includes open source code in which they hold copyright, they will check the legal notices for correct copyright statements that gives them credit for their work. If they don't find them, they set up the trap: They document all occurrences of license violations and inquire about one of them, using a traditional cease and desist letter. If the company signs the cease and desist declaration to get rid of the nuisance, the trap has sprung. The copyright troll will come back with other violations, asking the company to pay the typically hefty fines agreed to in the cease and desist declaration.

Patent trolls, more formally known as nonpracticing entities, don't have to set up a trap; they will simply approach anyone using software that uses their patents and try to extract license fees, with possible escalation to the courts if the victim doesn't pay up. Sadly, the availability of open source legal notices has made it easier for patent trolls to determine whether a particular software uses open source code they argue violates their patents.


Copyleft enforcers are organizations or individuals who might sue a distributor of open source code who does not follow any copyleft obligation in the distributed code. Typically, the open source code in question is the Linux kernel. There used to be individuals who tried to enforce copyleft licenses, but due to the legal costs of enforcement, most of this work has moved to the Software Freedom Conservancy, a U.S.-based 501(c)3 non-profit organization.

Historically, copyleft enforcement was driven by the moral impetus of "freeing all software" from the enslavement through software vendors. This heated rhetoric has abated, and today's primary goal simply appears to be license compliance: To respect the original open source developer's wishes as expressed by the chosen licenses.

The Software Freedom Conservancy does not randomly start lawsuits but rather approaches copyleft

enforcement strategically: It finds or is approached by copyright holders of copyleft-licensed code whose rights are being violated. It may then start work if the case appears suitable to move forward the goal of copyleft compliance. The suitability of a case depends on multiple factors, but perhaps the most important one is how representative of a category of violations the case is. Then, enforcing copyleft compliance may serve as a warning to others and prevent further copyleft violations in the first place.

A lawsuit is the last resort. First, the Software Freedom Conservancy will simply ask to have the copyleft violation be fixed. Over time, open source licenses have added cure periods during which the distributor can fix any license violations, for example, by providing proper legal notices or corresponding source code. These cure periods typically last 30 days but you may be able to agree on a longer period. Key is the obvious and honest willingness to solve the problem. The Software Freedom Conservancy is likely to move to a lawsuit only if the response is confrontational and the problem simply doesn't get fixed.

The situation keeps shifting. Copyleft enforcers, for example, have teamed up with the right-to-repair movement. To repair a malfunctioning device that runs on software, you obviously need the source code, whether it was built from copyleft-licensed open source code or not. As an outlook, we can expect more legislation to weigh in on the availability of source code. 

DIRK RIEHLE is the professor for open-source software at Friedrich-Alexander-Universität Erlangen-Nürnberg, 91058 Erlangen, Germany. Contact him at dirk@riehle.org.