January 11, 2023

FEATURE

The Internet of Things and the Challenges of Open Source Software Licensing

Aaron Williamson and Kate Downing

Share:

Internet of Things (IoT) devices—from Amazon Echos and Fitbits to the sensors that power smart cities—all contain embedded software. Most are built on the open source Linux operating system and contain dozens (or hundreds) of open source applications and software libraries. These open source components do everything from supporting the most basic functions of the device to providing the framework for the graphic interface you see when you log on to the device. (The rest of this article will refer to open source components collectively and individually as "OSS.")

IoT device makers don't cobble all of this OSS together themselves. Typically, they source their embedded computing hardware (or key portions of it) from original equipment manufacturer (OEM) hardware producers or system integrators, and these hardware suppliers provide a basic operating system for them. The device maker will then typically add their own custom software to provide user-facing functionality.

The many OSS components in an IoT device are all subject to copyright, and sometimes to patents as well. (1) They are made available subject to open source licenses that govern the reproduction, modification, and distribution of the software. The device maker must therefore comply with the licenses for all OSS embedded in its product, even if that OSS originates from a supplier, because it distributes all that third-party software every time it ships a product.

The presence of third-party OSS in an IoT device means that device makers must:

1. understand the terms of OSS licenses and assess whether they can comply with those terms;
2. provide end users the compliance materials required by the OSS licenses applicable to their products, including (a) copyright notices and license information for each OSS component and (b) source code for certain OSS components where required by the OSS license;
3. invest in the open source management tools, processes, and policies necessary to meet all these challenges comprehensively and efficiently; and
4. take steps to ensure that their suppliers provide them with the information necessary to enable them to fulfill their OSS-related obligations.

This article explores each of these challenges in further detail.

Open Source License Compliance

Open source licenses come in many shapes and sizes but can be broadly grouped into three categories.

Permissive licenses contain very few conditions. The MIT License, [2] for example, requires only that a copy of the license and the developer's copyright notice be included with all copies of the software distributed by the licensee. Permissive licenses (like most open source licenses) also typically include warranty disclaimers and limitations of liability. The Apache License [3] is unique among the most popular permissive licenses in its treatment of patents. It contains an express patent grant and provides for defensive termination of that grant if a licensee alleges that the covered software infringes its patents.

Reciprocal licenses [4] include an important quid pro quo. In addition to the requirements commonly found in permissive licenses, they require licensees to provide the source code for the OSS, including any derivative works, every time they distribute the OSS to a third party. By far the most common reciprocal license is the GNU General Public License (GPL) version 2. [5] Reciprocally licensed OSS requires the most care: if an IoT device maker incorporates a GPL-licensed component into a proprietary program embedded in its product, it may be obligated to provide the source code for the entire combined work to its customers.

Semi-reciprocal licenses (or "weak copyleft" licenses), such as the Mozilla Public License [6] and Eclipse Public License, [7] take the middle path. Licensees must provide the source code for the OSS itself, including any modifications they've made to it, to anyone they distribute the OSS to. But, they are not generally required to provide the source code for proprietary works that interact with or incorporate the OSS. [8]

Common Compliance Requirements

Any device of reasonable complexity will contain software under each of the three types of licenses outlined above. Therefore, every device maker can expect that its compliance obligations when distributing its product to consumers will include:

- Providing a copy of the license for every OSS component embedded in its product.
- Providing the copyright notices and other intellectual property (IP) attribution statements required by various licenses. Some licenses, like the MIT and BSD licenses, incorporate copyright statements into the license itself. This makes compliance simultaneously simpler (license and attribution requirements are met by providing a single document) and more complex (it's not sufficient to provide a single copy of the MIT License to cover all MIT-licensed components in your product, because each contains a copyright statement that must be included).
- Providing (or offering to provide) a copy of the source code for reciprocally or semi-reciprocally licensed components, and for any derivative works of the reciprocally licensed components.
- Including notices of its modifications to OSS components, where required. Many OSS licenses of all types require that, when a licensee distributes the source code for an OSS component, it must include notices in any modified source code files. So where a device maker is required to distribute the source code for a component (e.g., because it's reciprocally licensed) or chooses to do so, it must include these notices.

A minority of OSS licenses require licensees to include specific notices in the advertising for their products if the advertising mentions the OSS product or the features it provides. Sometimes referred to as "obnoxious advertising clauses," such provisions are probably overlooked more often than they're honored but are nonetheless worth keeping an eye out for.

Reciprocal Licensing Issues

While all open source licenses place basic conditions upon licensees when they distribute OSS, reciprocal licenses deserve particular attention, because they can impact the licensing of proprietary software components as well. Compliance with reciprocal (and even semi-reciprocal) licenses is a complex topic, and application is always case-specific, so we'll only cover the basics here. (9)

GPLv2, Copyleft, and Derivative Works

Version 2 of the GPL (GPLv2) is by far the most common reciprocal OSS license and the most commonly encountered in IoT software stacks. GPLv2 (like other variants of the GPL) has an ideological bent that's stated plainly in the license: its purpose is to ensure that the users of software have the freedom to study, modify, and redistribute the software that they use. It's a long license, partly because it endeavors to close many loopholes by which licensees might attempt to deprive end users of these freedoms. And its language is idiosyncratic: a lawyer can have years of experience reading and applying the license and still get stumped on occasion.

GPLv2 is also the license for Linux, the core of the operating system run by nearly every IoT device, and for a number of other core OSS operating system components that are more or less essential when building an embedded device. It also applies to some components that are commonly used in IoT devices, depending on their function. For example, FFmpeg, a library for video playback and manipulation, is subject to either GPLv2 or the Lesser General Public License version 2 (LGPLv2), depending on how it's configured.

The key provision to pay attention to in GPLv2 and other reciprocal licenses is the copyleft provision. In short, this term says:

1. if you distribute the GPLv2-licensed OSS to someone, you must also provide them with the complete corresponding source code;
2. if you distribute a modified version of the OSS to someone, you must provide the complete corresponding source code for the OSS together with the modifications; and
3. if you distribute software that is a derivative work of the OSS to someone, you must provide the complete corresponding source code for the entire derivative work.

What is a derivative work of a software application or library? That's a topic of vigorous debate and occasional rancor within the open source community, but there are a couple points of general agreement:

- When you modify the source code of an OSS component directly, you have created a derivative work and must include your modifications when providing source code to end users.

- If your software interfaces with an OSS program using its standard external interface (e.g., by running the OSS program and sending commands to it via the operating system), and you have not modified the OSS program, your software is not a derivative work of the OSS program.

The points of contention are far more numerous and stickier.

If your program depends upon an unmodified OSS software library, is the program a derivative work of that library? Does the technical means by which your program interfaces with the OSS software library affect the answer? Specifically, if your program is written in the C programming language, does it matter whether the program is "statically" or "dynamically" linked to the library? Static linking involves compiling both components together into a single executable program file. Dynamic linking involves compiling them separately but incorporating a small amount of interface code from the library into your program. Dynamic linking makes it possible for multiple programs on a system to use the library without duplicating its object code in each program. The functionality of the resulting program is the same regardless of which method of linking is used.

The Free Software Foundation—the nonprofit that authored and stewards the GPL licenses—takes the view that linking of either kind yields a derivative work. [10] This view is embedded in its semi-reciprocal license, the LGPL, which requires licensees to provide source code for programs that use the licensed library, unless the library is linked dynamically and the program incorporates no more than a minimal amount of interface code from the library itself. [11]

If you modify an OSS software library to include an external interface and interact with it via the operating system, specifically to avoid linking it to your program, is your program a derivative work of the modified library? This question was the subject of recent litigation by the Software Freedom Conservancy against VMware, which for many years used interface code from the Linux

kernel in its ESXi Hypervisor software, to enable ESXi's proprietary operating system kernel to interface with GPL-licensed device drivers. [12] The case was dismissed on grounds unrelated to the core claims, and was abandoned by the Software Freedom Conservancy when VMware subsequently removed the Linux code from its product. [13]

If you produce a loadable Linux kernel module containing a custom driver for hardware specific to your device, is that module a derivative work of Linux? Prominent maintainers of Linux believe that at least some kernel modules are derivative works of Linux. [14]

While several of these questions have been the subject of litigation, none have been definitively settled by the courts.

GPLv3 and Installation Information

Version 3 of the GPL (GPLv3) works much like GPLv2 but includes a number of new provisions. The most important of these for IoT device makers is the "Installation Information" provision in section 6. According to this term, if you include a GPLv3-licensed component in a device sold to consumers (a "User Product"), and it is possible to update the software in that device, you must include "Installation Information" as part of the complete corresponding source code for the GPLv3 component:

> "Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made. [15]

To paraphrase, the device maker must provide whatever information is necessary to enable an end user who has modified the GPLv3 component to install and run that modified copy onto the device. If installation of software onto the device requires the use of a special process or a cryptographic key, these must be provided to the end user. And the device can't be designed to cease functioning simply because the GPLv3 component has been modified (although the manufacturer can void warranties for modified devices and prohibit them from accessing its networks).

This provision caused a stir among consumer electronics manufacturers when it was introduced and was one reason among many that the maintainers of Linux have cited for not adopting GPLv3 and instead continuing to use GPLv2. Device makers still commonly avoid incorporating GPLv3 components in their products. While GPLv3 is less widely used than GPLv2, there are a number of OSS projects using GPLv3 that are potentially useful in embedded devices, including Samba (an application that enables Linux devices to interact with Windows systems) and most software produced by the GNU Project.

Compliance Materials

When an IoT device maker ships its product to an end user, it must also provide the compliance materials required by the applicable OSS licenses, including the OSS license texts, author IP notices, and source code for reciprocally and semi-reciprocally licensed components. This section describes common practical approaches to these obligations.

Licenses and IP Attribution

IoT devices distributed to end users must include copies of OSS licenses and IP attribution statements for each OSS component they contain. Product companies typically follow either or both of the following strategies for complying with the license and IP notice obligations:

- Printing all of the required license and attribution statements in the product manual or another printed document accompanying the product. (If the next IoT device you buy comes with a thick user manual, you can bet it's 75% OSS compliance information.)
- Compiling all of the required information into a single digital text and incorporating it into the interface of the device. (If you navigate to the "Legal" option in the menu for your smart TV, you will likely have the pleasure of scrolling through the full text of dozens of OSS licenses.)

It is *not* sufficient to include a link to a web page containing the required notices; most OSS licenses are clear that the information must be included with the (in this case, embedded) software when it's distributed. [16]

Source Code

For reciprocally licensed OSS, in addition to providing end users with OSS licenses and IP attribution statements, device makers must also include either a copy of the source code for the OSS or (depending on the license) an offer to provide the source code or information about how to obtain it.

Product companies generally prefer not to include the required source code for reciprocally licensed OSS components with the product. [17] Instead, most include a link or reference in their printed notices to a location on their website where they provide the required source code for their products. The source code will usually be contained in a compressed archive file (e.g., zip) containing a copy of the printed compliance notices and individual archive files with source code for each reciprocally licensed component in the product.

The source code provided must be the actual source code to the component embedded in the product—what GPLv2 refers to as "the complete corresponding machine-readable source code." [18] If the device maker has modified a GPL-licensed component, it's not enough to include the unmodified source code of that component as distributed by the open source project—it must include the modifications. For GPLv2-licensed components, the "complete corresponding . . . source code" also includes "any associated interface definition files, plus the scripts used to control compilation and installation of the executable," [19] meaning the code necessary to enable someone to configure and compile the code in the same way the manufacturer did.

Open Source Management

To comply with their obligations under OSS licenses, device makers must maintain accurate and comprehensive information about which open source components (and which version of each) are used in their devices. This requires significant effort because, as stated above, most IoT devices contain dozens of OSS components, and each of those may depend upon anywhere from a handful to more than a hundred other OSS components (which are sometimes referred to as "transitive dependencies").

License compliance is not the only reason to invest in managing this information—it's also essential to keeping the devices secure. While OSS is not inherently more or less secure than closed or proprietary software, it contains bugs just as all software does. Some of those bugs can make the OSS—and any device it runs on—vulnerable to attack by malicious hackers. To keep their devices reasonably secure, device makers must track information about vulnerabilities in

their OSS dependencies, fix newly discovered vulnerabilities as soon as they're discovered, and issue those fixes to end users promptly via software updates.

Finally, maintaining accurate OSS-related information is key to demonstrating proper controls to auditors, investors, partners, and customers (especially if the customers are large distributors, resellers, OEMs, or enterprises). Such information can also be vital both in the decision to bring a legal action against a third party [20] and in defending an OSS-related legal action.

Open Source Management Tools

Companies increasingly use automated tools to track this mountain of information. A growing industry provides software that enables companies to discover and track the OSS components in their products, provides updates about newly discovered security vulnerabilities in those components, and helps companies in preparing OSS license compliance information for distribution to end users. These tools integrate with a company's software development pipeline to automatically capture new and updated OSS components as developers add them to their software.

Processes and Policies

OSS management tools are not magic bullets; effective OSS management requires institutional knowledge and human intervention. Like all software companies, device makers are well served by creating written policies describing, among other things:

- What OSS management tools they use and how those tools work
- Who is responsible for maintaining such tools
- How they ensure that new software is scanned by or integrated with any OSS management tools
- Who is responsible for reviewing the results of such tools and at what cadence
- The process for dealing with an OSS component with insufficient information or a license that the device maker cannot comply with

- Who is tasked with putting together the license and IP attribution notices and source code described above, and the form in which this is done

- The creation, approval, and maintenance of a list of licenses that the device maker can and cannot comply with (a "go/no go" list) [21]

Device makers often utilize more than one "go/no go" list for their devices. It is common to have one governing the operating system layer of the device, which commonly contains a number of GPL-licensed components and is licensed from a supplier, and a separate "go/no go" list for application layer software typically written by the device maker itself. While device makers wishing to take advantage of Linux can't avoid the GPL at the operating system level of their software stack, they may restrict reciprocal and semi-reciprocal licenses at the application layer to ensure that product-differentiating applications remain proprietary while avoiding tricky analyses of software interactions. [22]

Relationships with Suppliers

When it comes to OSS license compliance, the venerated "garbage in, garbage out" rule applies: if a device maker doesn't get useful information about OSS components from its OEM hardware suppliers, its hopes of discovering all of the OSS in its product are slim, and its risk of compliance or security issues goes up. Even the best open source management tools can't effectively analyze software delivered to the device maker in binary form. And supplier-provided source code can pose its own challenges for such tools if the supplier's developers did not utilize proper coding hygiene. For these reasons, it is essential to gather this information directly from the supplier rather than to try to assemble it forensically.

Supplier Agreements

Device makers should also be sure to include appropriate provisions in their contracts with suppliers to address OSS license violations caused by the supplier. Since any violation of an OSS license is fundamentally copyright (and sometimes patent) infringement, device makers should look for broad warranties and indemnities related to IP infringement arising from supplier-provided software.

OSS-specific provisions may also be warranted when:

- The device maker wants to see OSS-related information presented in a specific format, preferably such that it is ready to be viewed by end users without further editing.
- The device maker has a "go/no go" list it wants suppliers to follow, particularly if the supplier is providing software intended to be used as a component in a proprietary piece of software in the application layer of the product.
- The supplier is legally unsophisticated, and it is useful to spell out what compliance with OSS licenses means.
- Failure to comply with OSS licenses could impact the device maker's relationship with the public at large or specific partners, customers, potential employees, etc., in addition to raising legal liability.
- The supplier is unwilling to offer broad warranties and/or indemnities but is open to specific representations, warranties, or indemnities solely with respect to open source. (23)

The specific provisions that may be warranted in any given transaction will also depend on the nature of the software the supplier is providing. Custom software, particularly software bound for the application layer of product (as opposed to the operating system), may naturally result in a lot of feedback and iteration between the device maker and the supplier. In such a relationship, suppliers are more likely to agree to seek the device maker's preapproval of particular OSS components, or to be amenable to following a specific policy regarding the OSS components to be used.

A provision like the following may be appropriate to ensure that a supplier does not incorporate reciprocally licensed OSS in proprietary application-layer software:

> Supplier will not incorporate into any Supplier Software any third-party software that does, will, or would reasonably be expected to require the (a) disclosure or distribution of the Device Maker Software in source code form, (b) licensing or other provision of the Device Maker Product on a royalty-free basis, or (c) grant of any patent license, non-assertion covenant, or other rights under any intellectual property rights or rights to modify, make derivate work based on, decompile, disassemble, or reverse engineer the Device Maker Product. Without limiting the foregoing, Supplier Software will not include

> any software licensed under any version of the GNU General Public License, the GNU Affero General Public License, or the SleepyCat License.

Note, however, that for devices that depend upon Linux, such a provision cannot be applied to the entire product (i.e., both the operating system and application layers), since Linux contains a number of reciprocally licensed components.

A provision like the following can be used to ensure that the device maker receives all necessary OSS-related information:

> During the Support Period, Supplier will:
>
> (a) provide Device Maker with a listing of all third-party software components that Supplier includes directly or indirectly in or with the Supplier Software upon delivery of each release thereof to Device Maker. Such listing will include for each component: (i) the name, (ii) the version, (iii) the download URL, and (iv) the full license text(s) as included in the component as well as any copyright notices. The listing will be of sufficient quality to be customer-facing without modification by Device Maker, and Device Maker may use and distribute the listing verbatim, with modification, or aggregated within other Device Maker documentation; and
>
> (b) provide Device Maker with all source code that Supplier is required to disclose under any applicable third-party software licenses.

Note that this provision puts the supplier on notice that its obligations extend to transitive OSS components ("directly *or indirectly*"), and it also extends all the OSS-related obligations to each and every release of software that the supplier may make to the device maker.

However, suppliers of "off-the-shelf" software (which typically includes operating systems) are less likely to agree to these sorts of provisions, and it becomes incumbent on the device maker to shop around for a trustworthy supplier. Device makers should look for a supplier that provides the licenses and copyright notices for the OSS components it uses, has made the source code easily available, and publishes information on its OSS compliance practices and policies. (24)

Conclusion

Open source software is an essential part of any IoT device, but it must be used mindfully to avoid security issues and ensure compliance with OSS licenses. Device makers should pick their suppliers carefully, adopt tools and processes to track the OSS components they use, and surface security vulnerabilities in a timely way. They should also develop institutional knowledge about OSS license terms, and engage outside expertise where needed, to ensure their use of OSS is consistent with the OSS license terms and with their business strategy.

Endnotes

1. Some third-party components or OSS subcomponents commonly used by IoT devices are dedicated by their authors to the public domain rather than released under an OSS license and are free for general use.

2. *The MIT License*, Open Source Initiative, https://opensource.org/licenses/MIT (last visited Nov. 16, 2022).

3. *Apache License, Version 2.0*, Apache Software Found. (Jan. 2004), https://www.apache.org/licenses/LICENSE-2.0.

4. Reciprocal licenses are also referred to as "copyleft" licenses (by supporters) or "viral" licenses (by detractors).

5. *GNU General Public License, Version 2*, GNU (June 1991) [hereinafter *GPLv2*], https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html.

6. *Mozilla Public License Version 2.0*, Mozilla, https://www.mozilla.org/en-US/MPL/2.0 (last visited Nov. 16, 2022).

7. *Eclipse Public License - v 2.0*, Eclipse Found., https://www.eclipse.org/legal/epl-2.0 (last visited Nov. 16, 2022).

8. An important caveat to this general characterization is the GNU LGPL, which is among the more popular semi-reciprocal licenses. The LGPL's source-provision requirement is discussed in detail below.

9. For deeper treatment of more specific issues, see Heather Meeker, Open Source for Business: A Practical Guide to Open Source Software Licensing (2d ed. 2017); Van Lindberg, Intellectual Property and Open Source: A Practical Guide to Protecting Code (2008); Armijn Hemel & Shane Coughlan, Practical GPL Compliance: A Guide for Startups, Small Businesses and Engineers (2017), https://www.linuxfoundation.org/tools/practical-gpl-compliance; Copyleft.org, Copyleft and the GNU General Public License: A Comprehensive Tutorial and Guide (2018), https://copyleft.org/guide.

10. *See Frequently Asked Questions about the GNU Licenses*, GNU, https://www.gnu.org/licenses/gpl-faq.en.html#GPLStaticVsDynamic (last updated Dec. 28, 2021) (refer to entry "Does the GPL have different requirements for statically vs dynamically linked modules with a covered work?").

11. *See GNU Lesser General Public License, Version 2.1*, GNU § 5 (Feb. 1999), https://www.gnu.org/licenses/old-licenses/lgpl-2.1.en.html. LGPL used to stand for "Library General Public License," as it was designed to enable open source libraries to be used by proprietary programs. Free Software Foundation founder Richard Stallman changed the name when he decided that it was better for developers to use the GPL for libraries as well, so that developers wishing to use the libraries would have an incentive to release their code under the GPL as well. *See Why You Shouldn't Use the Lesser GPL for Your Next Library*, GNU, https://www.gnu.org/licenses/why-not-lgpl.en.html (last updated Jan. 2, 2022).

12. *See* Scott K. Peterson, *GPL Enforcement Action in Hellwig v. VMware Dismissed, with an Appeal Expected*, Opensource.com (Aug. 11, 2016), https://opensource.com/law/16/8/gpl-enforcement-action-hellwig-v-vmware.

13. Press Release, Software Freedom Conservancy, VMware Suit Concludes in Germany (Apr. 2, 2019), https://sfconservancy.org/news/2019/apr/02/vmware-no-appeal.

14. *See* Email from Linus Torvalds, Open Source Dev. Lab, to Linux Kernel Mailing List (Dec. 4, 2008, 03:11 GMT), https://yarchive.net/comp/linux/gpl_modules.html.

15. *See GNU General Public License, Version 3*, GNU § 6 (June 2007), https://www.gnu.org/licenses/gpl-3.0.html.

16. *See, e.g.*, *The 3-Clause BSD License*, Open Source Initiative, https://opensource.org/licenses/BSD-3-Clause (last visited Nov. 16, 2022) (requiring that "[r]edistributions in binary form must reproduce the . . . copyright notice, . . . list of [license] conditions and . . . [warranty] disclaimer in the documentation and/or other materials provided with the distribution").

17. This was a fairly common practice back when most hardware products came with a physical CD of accompanying software, but gone are the days.

18. *GPLv2*, *supra* note 5, § 3(a).

19. *Id.* § 3.

20. For example, a device maker may wish to avoid bringing suit against the copyright holder of an OSS component that's essential to its product, as many OSS licenses contain termination provisions triggered by such claims (including Apache 2.0, Eclipse Public License 2.0, and Mozilla Public License 2.0).

21. The Linux Foundation's OpenChain Project publishes a standard for open source license compliance practices, which contains more detailed information about best practices for open source management. *See* OpenChain Project, https://www.openchainproject.org (last visited Nov. 16, 2022).

22. It is generally accepted that the GPLv2 license applicable to Linux does not affect the licensing of software at the application layer. *See, e.g.*, Karim Yaghmour, Building Embedded Linux Systems app. C.1 (2003) (noting that Linus Torvalds added a preamble to the Linux kernel license to avoid any confusion regarding the status of applications running on top of the Linux kernel), http://etutorials.org/Linux+systems/embedded+linux+systems/Appendix+C.+Important+Licenses+and+Notices/C.1+Exclusion+of+User-Space+Applications +from+Kernel+s+GPL.

23. Suppliers are often more comfortable with agreeing to comply with specific processes and/or deliverables than they are to represent they "comply with all relevant OSS licenses" because, per the discussion above, there remain a number of open questions with respect to various provisions of many licenses, particularly the GPL family of licenses.

24. The OpenChain Project publishes a specification for open source compliance programs that suppliers can adhere to and claim conformance with. Note that the specification ensures that certain artifacts documenting processes and controls exist, but the specification is agnostic as to "go/no go" policies, so mere conformance with OpenChain may not be sufficient to meet the needs of a specific device manufacturer. OpenChain itself also does not conduct any audits. However, claims of OpenChain conformance constitute one indication that a supplier is sophisticated in this area.

ENTITY:

SECTION OF INTELLECTUAL PROPERTY LAW

TOPIC:

INTELLECTUAL PROPERTY

Authors

Aaron Williamson
Williamson Legal, PLLC

Aaron Williamson is the founder of Williamson Legal in St. Petersburg, Florida. He counsels software companies and tech-focused nonprofits on technology transactions, open source issues, and restorative conflict resolution.

Kate Downing
Law Offices of Kate Downing

Kate Downing is the founder of Law Offices of Kate Downing in Santa Cruz, California. She counsels companies on implementing open source compliance strategies, technology transactions, and IP due diligence.