NOVEMBER/DECEMBER 2021

DEVOTED TO LEADERS IN THE INTELLECTUAL PROPERTY AND ENTERTAINMENT COMMUNITY

VOLUME 41 NUMBER 10

THE Licensing Journal®

Edited by Gregory J. Battersby and Charles W. Grimes

Wolters Kluwer

Open-Source Software: Risks and Rewards

Douglas Crisman, Janice Logan, Ph.D., and Manu Bansal, Ph.D.

Douglas Crisman, a partner with Morgan Lewis, brings the perspective of a software designer and intellectual property (IP) director for a leading computer hardware company to his patent law practice, which includes patent preparation, licensing, and prelitigation opinions, as well as IP transactions, due diligence, and counseling. He routinely works with standards-setting bodies and consortia on IP issues, and provides advice on strategic IP management and open source legal issues ranging from software development to code review and licensing, and open source compliance.

Janice (Lee) Logan assists clients with worldwide intellectual property (IP) strategies at Morgan Lewis. As a co-leader of the firm's Asian Life Sciences Working Group, she works with non-IP lawyers and brings a deep background in science to the firm's life sciences practice, focusing primarily on chemistry, biotechnology, and medical devices. Janice guides clients through complex patent procurement and patent litigation matters. She also manages due diligence for IP asset transactions. Janice is fluent in Korean and Japanese.

As a seasoned patent law professional with over a decade of experience and using his strong technology background, Manu Bansal assists clients at Morgan Lewis in building and managing quality patent portfolios attuned to their business goals. He prepares and prosecutes patents in electrical and computer engineering areas including video coding, wireless communication, network architecture, software, semiconductors, and artificial intelligence. Manu works on strategic prosecution and post-grant proceedings involving standard-essential patents directed to video coding standards (HEVC/H.265, VVC, AV1, VP9) and wireless standards (4G/LTE, 5G, IEEE 802.11). He is admitted in Virginia only, and his practice is supervised by DC Bar members.

For companies that develop software, whether for use on internal servers, as part of a cloud-based business model, or for distribution to end users, open-source software (OSS) offers some compelling advantages, including low cost and wide availability. However, before committing to any OSS solution (especially in a mission-critical role), a business needs to ask whether the benefits justify the risks.

This article discusses some of the benefits and risks of open-source–based software development, including the risks inherent in trusting business functions to public code and in mixing OSS with company code. Also addressed in this article is a process for avoiding the unintended and unwelcome consequences of an ill-considered commitment to a particular OSS solution.

OSS: An Overview

The concept of OSS originated in the "free software" movement in the early 1980s, which was a reaction to concerns the software community had with the proprietary software model at that time. Under the proprietary software model, an end user is provided an executable program, but the end user is dependent on the software developer or provider for bug fixes, upgrades, and general maintenance of the software. To address these concerns, the goal of the free software movement was to obtain and guarantee freedoms for software users including, for example, freedoms to run, study, copy, distribute, and modify the software.[1]

The free software movement brought the formation of the Free Software Foundation (FSF), which supported the GNU Project[2] launched in 1984 to develop a free, UNIX-like operating system, commonly known as Linux. Linux was released under the GNU General Public License (GPL), also famously known as the "copyleft" license, which was so named to signal the freedom to modify or distribute software that this license provides, in contrast with the restrictions for such activities inherent in traditional copyrights. The GPL copyleft license was drafted to ensure that free software, including any modifications to the software, remains free (not proprietary).[3]

However, at the time, the free software movement was generally viewed as anti-business, and corporations were concerned about the new model permitting developers to freely copy, modify, or distribute software, or essentially exercise the rights normally protected under copyright law, with the only obligation being to distribute to others under the copyleft license. These views started turning in 1998 when Netscape announced that it was considering sharing the source code of its browser, and corporate America began to buy into the OSS movement. The OSS movement was organized into a nonprofit corporation, the Open-Source Initiative (OSI).[4]

The OSI sought to quickly establish itself as the "gold standard" of open-source licensing by publishing the "Open Source Definition," which provides clauses to define attributes that software must embody in order to be considered open source and outlines distribution terms of OSS.[5] Defined simply and broadly, OSS is source code that may be freely shared with other programmers, subject to an open-source license. The use of OSS is now ubiquitous and has seen explosive growth in recent years. For example, per Synopsys, there were 84 open-source components per commercial application in 2016, and that number grew to 528 open-source components in 2020.[6] Some of the most famous and widely used OSS packages include Linux (operating system; GPL v2), Apache (web server; Apache License 2.0), MySQL (relational database; GPL v2), Perl (scripting language; Artistic License and GPL v2), OpenStack (cloud-computing platform; Apache 2.0), Apache Hadoop (framework for big data; Apache 2.0), and R (statistical computing language; GPL v2).[7]

OSS Benefits

Some of the benefits of OSS include rapid deployment and low cost. For example, instead of spending months or years developing an application, a developer may easily access and download (for very low cost or free) relevant source code from open-source platforms such as GitHub and begin modifying it for his or her application. As such, the OSS remains available, modifiable, and maintainable. OSS can also be reliable and secure because, generally, a group of developers is constantly monitoring and assessing the source code for bugs and security holes, and reporting and fixing the issues to continually improve the overall quality of the software. Another advantage is the community that is built around the development of OSS. For example, OSS is built, developed, and maintained by many developers and promoters, and, as such, the contributors enjoy a pride of ownership in the software that is available to a larger community for free.

The structure of OSS development also lends itself to a unique peer development and partnership model, as the contributors may be individuals, nonprofit organizations, and corporations. Another benefit to corporations from relying on an OSS platform is the ability to outsource one or more portions of an application that they are developing to the open-source community so that they can focus on building and integrating the proprietary portions of the application. Overall, OSS provides an open standard that is commonly developed, improved, and maintained for compatibility by many users and entities.

OSS Risks

Using OSS involves significant risks, which fall into two broad categories—risks related to using open-source code instead of proprietary code and risks related to open-source licenses. The following is a brief overview of these and some other potential issues.

- Open-Source Code–Related Risks

Unlike source code for proprietary software, the provenance of the source code of the OSS may be unknown. For example, the OSS may include source code, the origins of which may be untraceable, or may include source code of a third-party proprietary software. Further, typically there is no formal support or warranty for the OSS. Also, in some cases, even though an OSS product may be widely used, the development efforts for that software product may be poorly funded, which can lead to poor software maintenance. Accordingly, a user of such an OSS product may not be able to rely on it for issue-free deployment and execution to the same extent that the user may rely on a proprietary software product.

As discussed above, OSS can be reliable and secure because any issues or security holes are constantly identified and addressed by a wider group of developers. The flip side of this is that its vulnerabilities are also open to the public and, as a result, it may be susceptible to significant security risks. Further, the development of OSS may be out of sync with the needs of a software company that is intending to use the OSS. For example, the company may not be able to have specific bugs fixed or features added to the OSS distribution, or may have an internal roadmap with OSS dependencies that does match the continual development path of that OSS. Also, since it might not have any role as developer of the OSS, the company may have no control or predictability as to the development or quality of its own software product that relies on OSS. Also, if a company applies modifications or customizations to a version of OSS, which is free to do, it may need to apply the modifications or customizations to every new version of the OSS, which may not be practical.

Another risk to consider is the mingling of the proprietary code with the open-source code and vice versa, which may raise challenges in licensing the proprietary software. For instance, under some open-source licenses, if an entity develops software that includes OSS, it is required to license that software under the open-source license, which means granting recipients a license to copy, modify, and redistribute the software for free. Thus, the developer entity may lose out on any licensing revenue from what it considered or intended to be proprietary software. Moreover, in situations in which the software developer is requested to disclose the source code, it may not want to or may not be able to untangle the proprietary code from the open-source code.

- Open-Source License–Related Risks

The mixing of proprietary code with open-source code may result in unwarranted licensing complications. Specifically, under a copyleft open-source license (*e.g.*, the GPL), the distribution of the software that has open-source code integrated with proprietary code could (based on the nature of the integration) trigger the obligation of the software developer to disclose the entire source code, including the proprietary code, under the copyleft open-source license terms.

Open-source licenses are generally non-negotiable, *i.e.*, to be accepted as is without any flexibility in modifying license terms. Also, not all open-source licenses come with the same or similar scope, and, in fact, many open-source licenses include non-standard terms that activate only under specific circumstances, thus changing the scope of the license. An example of this is the GPL, under which it is completely safe to use or modify the OSS for internal use, but, if the software is distributed, the obligation to disclose or share the entire source code is triggered, which may be a concern in certain conditions.

Another example of unique open-source license terms relates to patents. Some open-source licenses include patent-related provisions under which certain uses of the OSS may impose the obligation on the developer/user of the OSS to grant patent licenses to others for free. Such patent-related terms may trigger obligations to grant licenses to patents currently owned by the OSS user or may be even more burdensome by asking the developer to grant licenses to future patents.

The language of open-source licenses may be ambiguous and, if it has not been litigated, it may be unpredictable as to how it would be construed or interpreted if litigation ever resulted.[8] Also, if there is a contentious situation alleging violation of the terms of an open-source license, it may not be possible to resolve the issue privately by way of negotiations between the opposing parties.[9] Rather, the enforcement efforts can become public, and can create reputational issues in addition to legal issues for the entity alleged with the non-compliance of the open-source license.[10]

Accordingly, for any developer aiming to acquire and use or incorporate OSS, it is important to carefully review and understand the terms of the open-source license and consider whether the license terms are consistent with its intended use and the developer's ability to comply with the terms.

Cautionary Tales from Use of OSS

The following are two case studies that illustrate how some uses of OSS created problems for corporations and the lessons we can learn from them.

- Case Study 1—Hyper-V (2009)

Here, Linux driver code (under GPL v2) was apparently incorporated into proprietary Hyper-V Linux driver code. This usage of open-source Linux driver code was discovered when a user of the Hyper-V driver code reported, on a Linux internet blog, that "[t]he driver had both open-source components which were under GPL, and statically linked to several binary parts." Following publication in the open-source community of this alleged GPL compliance issue, the licensor of the proprietary driver code reacted swiftly by releasing its Hyper-V drivers as OSS under the GPL. Some lessons that can be learned from this case are:

- Lesson 1: Training for coders and developers on proper usage of OSS is important! It's not clear how the GPL driver code came to be incorporated in the Hyper-V driver, but perhaps this could have

been avoided through additional coder training (*e.g.*, on company policies regarding the use of and access to open-source code) and code review focusing on the use of OSS.
- Lesson 2: Alleged failure to comply with terms of an open-source license can generate unwelcome news in the open-source community. To avoid such news, companies need to understand the requirements of the open-source licenses they use and be ready to address compliance issues.
- Lesson 3: Developers of proprietary code may want to consider whether to allow coder access to OSS licensed under copyleft licenses.

- Case Study 2—Heartbleed Bug (2014)

This case highlights potential security risks of relying on open-source code. In this case, there was a bug in OpenSSL, which is a widely used open-source toolkit used to provide secure communications between web clients/browsers and websites. The bug allowed passwords to be captured and affected nearly two-thirds (!) of internet users (excluding banks and government entities). However, the bug was publicly disclosed at Openssl.org and was fixed shortly thereafter.[11] Some lessons that can be learned from this case are:

- Lesson 1: Ubiquitous OSS components can be vulnerable and can impact software security for a large number of users.
- Lesson 2: The open-source community can be relied upon for discovery, disclosure, and fixing of code vulnerabilities. For example, here, the OpenSSL community was transparent about the bug and released a fix the same day as it was discovered and announced.
- Lesson 3: It is important to review the level of support and resources dedicated to key open-source projects. In this case, in 2014, the OpenSSL project, which was used by thousands of companies, reportedly had only one developer, who was earning no more than $2,000 in donations each year.[12]

Trends in Open-Source Projects

Recent trends show that many interesting open-source projects cover a wide variety of technical areas of interest including, but not limited to, big-data analytics, machine-based learning, cloud platforms, and blockchain. Some of the trending open-source projects include:

- Hyperledger by Linux Foundation—related to modular tools to promote commercial applications of blockchain technology
- Open Stack—a cloud operating system that allows vast computer, storage, and networking resources to be provisioned and controlled through a user-friendly dashboard
- R programming language—a popular open-source tool for data manipulation, calculation, and graphical display
- Life sciences—an extensive library of open-source tools available from institutions such as the Fred Hutchinson Cancer Research Center

Patents and OSS

In some respects, an open-source license can render patents related to the OSS unnecessary or ineffective. For example, distribution or other public use of OSS may carry an implied, or, in some instances, an explicit, license to patents covering the functionalities of the OSS.

However, patents may still provide a strong shield to protect intellectual property surrounding OSS in situations where a competitor takes undue advantage of the OSS release. For example, in an attempt to circumvent the open-source license terms, a competitor may implement one or more functionalities of the released OSS in a proprietary software project without using the open-source code. In such cases, patents covering those software functionalities can still be enforced by pursuing a patent-infringement action against the competitor.

Is an Open-Source License a Contract?

The answer is likely yes. In *Artifex Software, Inc. v. Hancom, Inc.* (N.D. Cal. Apr. 25, 2017), the court found that the plaintiff adequately pled a breach-of-contract claim based on alleged violation of terms of the GNU GPL, *e.g.*, due to incorporation by the defendant of the GPL open-source code in the proprietary code. Additionally, the court also found that the plaintiff's contract claim would not be preempted by its copyright-infringement claims. Accordingly, companies should be mindful of the fact that misappropriating OSS and non-compliance with the open-source license terms can potentially expose them on two fronts—under a breach-of-contract claim and a copyright-infringement claim.

Licenses Targeted at Cloud Uses of OSS

Under many copyleft open-source licenses (*e.g.*, the GPL), the source code disclosure obligations are triggered only when a licensee *distributes* software that includes or is derived from the OSS. However, some open-source licenses impose source code disclosure obligations on some uses of the open-source code or its derivatives to offer software services to users via a network. For example, the Server Side Public License (SSPL) requires disclosure of source code of modified versions of the program, as well as ancillary code that supports the software, if a licensee enables third parties to interact with the functionality of the program via a network. The popular database software, Mongo DB, which is used on servers, has adopted the SSPL. As another example, Elasticsearch, a popular search software used in cloud applications, recently transitioned from the Apache 2.0 License to the SSPL. Also, Plausible Analytics' web analytics software transitioned in October 2020 from the permissive MIT License[13] to the AGPL v.3, which extends the source code requirements of the GPL v3 to "prominently offer all users interacting with [your version of the Program] remotely through a computer network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software."[14].

In view of these open-source licenses directed to cloud-based applications, it is important for open-source users to be alert about license changes and evaluate risks associated with planned uses of the affected software.

Best Practices for Using or Contributing to OSS

The overall goal for an organization or company should be to promote safe use of OSS to leverage its benefits and mitigate its risks. Failure to effectively manage and track OSS use can result in violations that may remain undetected for a long time but come to surface at a critical, inopportune juncture (*e.g.*, during an acquisition or investment diligence process) when it may be difficult to correct the situation. Accordingly, it is imperative to establish a written open-source policy, as well as internal processes to implement and enforce the policy. For example, there should be set processes to review and approve OSS use requests and also track use of the OSS. Training programs for coders and developers should be established and regularly available to educate them on open-source licenses and their terms (*e.g.*, the point in software development at which the obligation to grant a license and disclose the source code attaches). In setting up a review process for open-source use requests, there may be different review tracks for different types of uses/licenses (*e.g.*, for strictly internal uses of unmodified OSS vs. OSS used in distributed code or a software package). For efficiency, a fast-track approval process may be considered for a limited set of licenses and/or a limited set of uses.

For careful consideration of open-source use requests, each request should identify various factors for the request, including, but not limited to, OSS version, its known vulnerabilities, all applicable licenses, availability of the same code under a non–open-source license, and the strength of the open-source community (to gauge the scope of the support and maintenance of the open-source code). Further, a request should lay out proposed uses of the open-source code. For example, is the requested use for a company product? Will the open-source code be modified? Will it be integrated with proprietary code and, if so, in what manner (*e.g.*, copy-paste, statically linked, dynamically linked, or API call)? Will it be server-based or used in a cloud/software as a service (SaaS) offering?

In terms of open-source use guidelines, some uses that are generally considered safe include using OSS under the Berkeley Software Distribution (BSD) or MIT license, running company code on Linux OS, using Lesser General Public License (LGPL) libraries without modification, and running OSS only on servers with no distribution (although beware of AGPL and SSPL licenses). Companies using OSS should be mindful of the risks involved in integrating any OSS with proprietary code. Moreover, a generous dose of caution may be warranted when developing non-GPL software that is compatible with the functionality of GPL software, allowing developers to use GPL source code, or accepting any third-party code for use in one of the developer's software products without understanding where it came from or under what license. Lastly, it is a good practice to check code dependencies and related licenses, as open-source codes can incorporate other open-source codes.

Companies considering contributing to OSS should invest in establishing an internal review process. Any review of the potential contributions should first consider reasons for contributing (improving functionality of strategic OSS, promoting wider use of company technology, adding customizations to an open-source project, outsourcing coding to the open-source community, and/or improving standing with the open-source community, press, and customers). Companies should also consider what license will apply for contributions, whether the contributions would be subject to third-party encumbrances, whether the contributions relate to any company patents, whether there would be a need for multiple source code trees in the future, and/or whether the contributions would harm the company's revenue.

When acquiring or investing in companies where software is a valuable part of the deal, it is important to conduct open-source due diligence. As part of the diligence process, the acquiring/investing company should ask the target company to identify (a) specific OSS items used by the target company, including OSS licenses associated with each item and each item's dependencies; (b) the context of each use, *e.g.*, whether the OSS is run on a company server or as part of an SaaS/Cloud offering, and/or distributed to end users/licensees; and (c) the extent of integration with proprietary code, *e.g.*, whether the target has made any modifications and/or contributions to the OSS. The acquiring/investing company may want to request a commercial software composition scan (*e.g.*, using the BlackDuck software) to identify license conflicts and request details as to how the target company manages OSS, trains employees to safely use OSS, and addresses OSS vulnerabilities.

1. *https://www.gnu.org/philosophy/free-sw.html*.
2. GNU stands for "GNU's Not Unix." *See http://www.gnu.org/gnu/gnu-history.html*.
3. *https://www.gnu.org/gwm/libredocxml/x53.html*.
4. *http://www.opensource.org/*.
5. *https://opensource.org/licenses/alphabetical*.
6. *https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html?intcmp=sig-blog-ossra1*.
7. *https://www.synopsys.com/blogs/software-security/top-open-source-licenses/*.
8. *https://www.gnu.org/licenses/gpl-violation.html*.
9. *https://sfconservancy.org/copyleft-compliance/*.
10. 2015 VMware Suit – unlicensed use of Linux code in proprietary "vmkernel" (*https://sfconservancy.org/news/2015/mar/05/vmware-lawsuit/*); 2019 Case dismissed (procedural grounds) - Vmware agreed to remove vmklinux from vSphere product (*https://sfconservancy.org/news/2019/apr/02/vmware-no-appeal/*).
11. "A missing bounds check in the handling of the TLS heartbeat extension can be used to reveal up to 64kB of memory to a connected client or server (a.k.a. Heartbleed)." "Fixed in OpenSSL 1.0.1g (Affected 1.0.1f, 1.0.1e, 1.0.1d, 1.0.1c, 1.0.1b, 1.0.1a, 1.0.1)." *http://openssl.org/news/vulnerabilities.html*.
12. *https://www.theregister.com/2021/05/10/untangling_open_sources_sustainability_problem/*.
13. A permissive free software license originated at the Massachusetts Institute of Technology (MIT). *See https://en.wikipedia.org/wiki/MIT_License*.
14. Section 3 of the GNU Affero General Public License.