SYNOPSYS®

WHITE PAPER

What Lawyers Need to Know About Open Source Licensing and Management

Matthew H. Jacobs, Esq., Director, Legal Counsel

Table of contents

Open source in commercial codebases

Modern software development involves assembling software parts and pieces from a variety of sources. These parts and pieces may be third-party commercial code, proprietary code, or open source code. If software development is like putting puzzle pieces together, open source represents an ever-growing and important part of that puzzle. So lawyers need to be aware of all the potential risks that come with using open source components in a commercial codebase if they are to advise their clients and protect them from those risks.

Each year, the Black Duck Audit Services team at Synopsys conducts open source audits on thousands of applications for its customers—primarily in conjunction with M&A transactions. Published in the annual Open Source Security and Risk Analysis (OSSRA) report, the latest analysis examines findings from the anonymized data of the thousands of codebases audited last year over hundreds of transactions.

More than 95% of the audited codebases in the OSSRA report contained open source.

As suggested by the results of the OSSRA report, the reuse of open source in software development is a good thing. Repurposing open source code allows software developers to move fast and avoid cost. By using open source in software development, organizations can perform three functions simultaneously:

- Reduce cost
- Develop better features
- Accelerate development timelines

Otherwise, it's difficult to improve all three aspects at the same time.

Costs of open source use

Although open source is free of acquisition cost, it is not free of management cost or risk. Critically, those reusing open source must confirm initially, and on an ongoing basis that:

1. The way they are reusing open source complies with the governing open source license.
2. The open source they're using does not contain any known security vulnerabilities.

A real challenge arises in managing these risks at an enterprise scale and pace. Developers are encouraged to reuse open source to do their jobs better, faster, and cheaper. However, management and legal visibility into what open source developers are using, as well as where and how they're using it, is often lacking. It follows that automated tools for identifying and tracking any open source being reused by an organization have become an indispensable part of a meaningful open source management program.

More than 85% of audited codebases had license conflicts.

Open source license compliance is a matter that many lawyers have been exposed to at some point, but for some, it remains shrouded in mystery. This is likely because there is little legal precedent in the field, the most popular open source licenses tend to be opaque, and the heavy use of acronyms creates a unique language that can handicap the uninitiated.

GNU General Public License (GPL)

Chief among open source licenses, and the license that garners the most attention, is the GNU General Public License (GPL). GPL 2.0 (GPLv2) is the license that governs, and will always govern, arguably the most important open source project in history: Linux. Owing in part to the popularity of Linux and the pioneering work of the Free Software Foundation in the field of open source, the GPLv2, and now the GPLv3, are critical mainstays of open source licensing.

The GPL is the source of much discussion concerning its interpretation, application, and overall impact on those who reuse software code licensed under it. The primary concern, shared by most lawyers, is that if a client/company reuses any code licensed under the GPL alongside, in connection with, or as part of any of the client's own code, the terms of the GPL will somehow apply equally to the client's code. In other words, by implication of the GPL terms, the client's code will be "infected" by the "viral" terms of the GPL.

More than 65% of audited codebases contained open source with no license or a custom license

There is some basis for this concern, but the GPL should not be read this way. First, the GPL freely allows internal use and modification of licensed code with no obligations. Companies are free to use Linux, or any other projects licensed under the GPL, for internal operations. They may modify those GPL-licensed projects as much as they like. And they can do so with no concerns about implicating any of the possibly unwelcome terms of the GPL.

Take care, however, when the client's intent is to modify GPL-licensed code, incorporate some of their own code, and then *distribute* the resulting work. It is the act of distribution, or conveying, GPL and client proprietary code that may implicate certain provisions of the GPL in a manner that may be undesirable by the client.

GPL and copyright law

Boiling it down, you can think of the GPL's so-called copyleft or reciprocal terms as a natural extension of basic copyright law. Copyright law extends a set of rights to the copyright holder, typically the original creator of some work (in this case, software). One of those rights is the right to grant others the ability to create derivative works. The GPL is essentially saying to anyone who accepts code under it (the licensee) that the copyright holder (the licensor) is granting the licensee the right to create a derivative but that if the licensee distributes that derivative, such derivative will likewise be subject to the GPL. Buried in this are questions that a lawyer must address:

- When has a derivative been created?
- What are the boundaries of that derivative?
- When does a distribution happen?
- What, precisely, are a licensee's obligations if it has distributed a derivative of code originally licensed under the GPL?
- What happens if the licensee fails to follow the obligations of the GPL, either intentionally or because it was unaware that GPL code had found its way into the licensee's development environment?

The answers to these questions can depend heavily on context. However, before engaging in a legal analysis of how the GPL might apply to a client's code, it is important to be clear whether the GPL necessarily applies to that code.

Implications of using GPL-licensed code

Ideally, a client who reuses GPL-licensed code will have considered all potential implications of their reuse before incorporating that code into their own software and then distributing the resulting derivative product. However, as discussed above, given the incredible amount of open source available, the pace at which it is used, and the frequent lack of management oversight, GPL-licensed open source inevitably and without the client's knowledge finds its way into software codebases that a client expected to keep entirely proprietary. In such event, according to the GPL, the undesirable outcome may be that the client's software likewise must be licensed under the GPL.

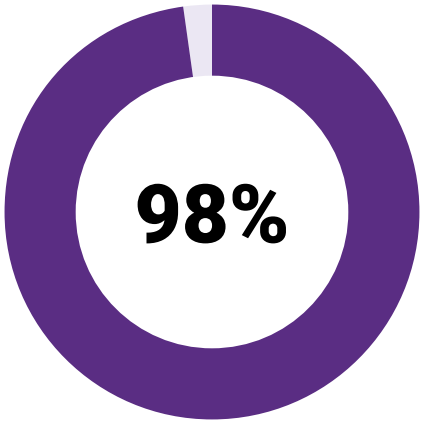
In a perfect world, the client will have identified at the outset any code they were working with that was originally licensed under the GPL and will have taken the necessary steps to use that code in a manner consistent with the obligations of the GPL and the client's goals. Failing this, remediation approaches include rewriting the code to remove the GPL-licensed code, approaching the code licensor (if one can be identified) and seeking permission to use the code under some other license, or, the riskiest option, doing nothing.

That last approach—do nothing and see whether anything comes of it—underlies a frequent sentiment: Many believe it unlikely that copyright holders will become aware of some misuse of their code licensed under the GPL, and even more unlikely that they will do anything about it. This sentiment, however, misses a critical element of open source reuse and management: the real-world business implications.

For instance, open source analysis is a staple in M&A and investment due diligence. Target companies, even those that are not primarily in software development, should expect an interested party to perform scans against their code to identify the following:

- What open source it contains
- What open source licenses apply to that code
- Whether the company is in compliance with those licenses

Here, it is not some copyright holder making the target pay for a lack of good open source management practices but rather the acquiring company using this fact as leverage to drive down a purchase price or to obtain more favorable deal terms.



Percentage of open source analyzed that was covered by the top 20 open source licenses

Permissive licenses

Another category of licenses, often referred to as permissive licenses, are easy to comply with and are free of the copyleft or reciprocal concerns of the GPL. The MIT and the BSD licenses, for example, are popular permissive licenses. Both essentially say that the licensee is free to use the code licensed under them in any manner that the licensee requires, provided the licensee gives attribution, essentially credit, to the licensor.

Given the open-ended nature of permissive licenses, code originally licensed under a permissive license is often relicensed under a closed source proprietary license or even repackaged and licensed under the GPL. So before you accept as fact that a piece of code is licensed under the GPL, it is worth investigating that code's pedigree to see whether its copyright holder originally released it under a permissive license. If you can determine that the code was originally licensed under a permissive license, you can choose to license it under that permissive license, instead of the potentially less attractive GPL, and follow the obligations of that permissive license.

Open source in the supply chain

Another key area where a lack of good open source management practices can cause financial pain is in supply chain management. Many customers require each vendor to produce a comprehensive list of all the open source and third-party software contained in that vendor's products. For vendors, being able to readily produce such a list, and stand behind it, is clearly a competitive differentiator. Vendors that can't do this may see a lost opportunity. Lawyers can help their clients gain this competitive edge by keeping tabs on the open source components in use and their applicable licenses.

Challenges of open source software use

To this point, we've focused on the challenge of managing vast amounts of open source and the potential license compliance–related issues. However, open source is software, and all software is susceptible to security vulnerabilities.

Open source is not any more or less secure than any other code, but again, it does present some unique challenges. Licensees of commercial software typically expect the licensor to remain aware of potential threats to its code and act quickly to inform the customer of issues and remedy those threats. After all, the licensor has a reputation in the market to protect. In sharp contrast, the need to keep alert to threats and vulnerabilities with respect to open source rests squarely on the shoulders of the users of that open source. Users of open source depend on their lawyers for advice on how to proceed in this scenario.

Open source vulnerabilities

For users of open source, staying apprised of what vulnerabilities have been identified and how to remediate those vulnerabilities is a task that can take different forms. On one end of the spectrum is a "keep your ear to the rail" approach. This hit-or-miss method requires monitoring news feeds, LISTSERVs, and perhaps the publicly available National Vulnerability Database (NVD) maintained by the U.S. federal government.

Nearly 100% of transactions included codebases that contained open source with at least one vulnerability, and almost 80% of transactions involved code with high-risk vulnerabilities.

On the other end of the spectrum is a more automated approach. Automated tools can identify, track, and monitor what open source an organization is using, compare that list in near real time to a variety of sources of open source vulnerability data, and send automatic alerts to designated individuals if the tools discover that the open source used by an organization has any vulnerabilities. These alerts can contain actionable information on how to address any identified vulnerabilities, whether temporarily or permanently.

Open source vulnerability disclosure

Vulnerabilities in open source tend to result from flaws in the way the code was written. These vulnerabilities can go undetected for years after the flawed code is introduced into an open source project. During that time, a popular open source project may be used and reused and incorporated into a wide range of other projects or deployed widely without the vulnerability being detected, or at least brought to the attention of those deploying or using the affected code.

Almost 100% of transactions included codebases with components that had no new development in the past two years.

At some point, typically as a function of direct or even unrelated research, a vulnerability is detected. When this happens, researchers are likely to make their findings publicly known. They may also post a description of how to exploit the vulnerability and how to patch it. Ultimately, this information may find its way into the NVD (although the process can be slow and sometimes nonoperational, like during a government shutdown). Importantly, this information is equally available to those who intend to do good as well as those who intend to do bad.

CVE-2017-5638, the Apache Struts vulnerability

An unfortunate and well-known case of an organization failing to pay proper heed to the potential vulnerabilities in the open source it was using is the infamous case of Equifax from 2017. Equifax used a popular open source web framework called Apache Struts. Lots of organizations use Apache Struts, and there is nothing inherently wrong or risky in doing so. However, when an exploitable vulnerability was discovered and disclosed in the version of Apache Struts that Equifax was using, Equifax failed to remediate it, leaving the personal information of millions of customers at risk.

Since hackers have access to the same vulnerability information as anyone else (or may have, through their own "research," identified otherwise unknown vulnerabilities), and since they are aware of which organizations are likely using what open source projects, organizations must put themselves in a position to know immediately when a vulnerability may impact them and to react quickly.

Real-world implications of open source use

Many lawyers act as gatekeepers; after evaluating any potential licensing challenges, they seem to remove themselves from the ongoing management of open source reuse. Given the implications of a security breach, this is a mistake. Lawyers must stay involved and keep driving the continuing management of open source if they are to act in their clients' or company's best interests.

Highlighting the real-world business implications and the security-related concerns surrounding open source management is not to say that there is no legal action in this area. There is. However, the nature of the action has morphed from enforcement for ideological reasons to enforcement driven by companies against other companies for commercial reasons. This company-versus-company enforcement is potentially more concerning for open source users, since it isn't just about "doing the right thing" but rather motivated by commercial drivers, such as lost revenue or competitive advantage.

Dual licensing

The classic example of commercially driven license enforcement is dual licensing. A company, as the copyright holder with the rights to license their software in almost any way they deem fit, may decide to license essentially the same code under two separate licenses. They may license their code under a traditional OEM-style license to anyone who wants to take that code and reuse it or embed it in a distributed commercial product without any GPL copyleft concerns. That same licensor may simultaneously license the same, or essentially the same, code under the GPL for those who want to use that code internally or who have no concern for the possible GPL copyleft implications, for whatever reason.

It happens with some frequency that a licensee, either out of lack of good open source management or negligence or as a product of willful misconduct, uses the code licensed under the GPL in their commercial applications where they should have paid for and used an OEM license that reflects their intended or actual use. In this scenario, it's also likely that the licensee is not adhering to the other requirements of the license and is in breach.

This is straight-up copyright infringement. The GPLv2, under which many of these cases arise, has no cure provision. Accordingly, any breach of the GPLv2 results in an immediate termination of all rights granted thereunder, and thus any continuing use of that software is unlicensed. The dual-licensing scenario presents an easy case for calculating damages because the licensor can look at the OEM side of its business and calculate what the infringing licensee should have paid in royalties had that licensee assumed the correct license in the first place. Dual licensing is not the only scenario in which company-versus-company enforcement arises, but it is frequent and illustrative.

Licensing *out* of open source

To this point, we've been discussing the licensing *in* of open source. However, many companies consider the licensing *out* of open source to be an important part of their business strategy. Companies adopting this strategy may include those who:

- Are engaged in dual licensing
- Wish to be open source friendly to attract and retain good engineering talent
- Want to use the power of open source to encourage the adoption of a new platform or set of tools or other functionality that it hopes to then sell services or add-on software around

The decision to license out one's own code, especially under one of the permissive-style licenses, should not be taken lightly. Once done, it is very hard, if not impossible, to reverse the implications of that decision. Recently, companies such as Redis Labs and MongoDB have expressed misgivings about building their businesses around permissive open source-licensed platforms in a manner that has allowed others, including competitors, to leverage those platforms without giving any commercial advantage back to the licensors (Redis Labs or MongoDB) or contributing any improvements or developments to the open source community generally.

Differences of opinion exist with respect to how a licensee who takes advantage of open source code exclusively for their own commercial advantage should be perceived, but the fact is that the permissive open source licensing model fully allows for this.

How to stay abreast of open source legal issues

Open source is an indispensable and critical element of modern software development. As the pace of software development continues to accelerate and the complexity of code continues to grow, the need for real-time management of open source assumes greater importance. Organizations and their legal counsel should think proactively about their processes and procedures for consuming open source code, and distributing products that use this code, in a manner that meets their business, security, and compliance goals. While this is a multifaceted challenge, the first step in this analysis should be to ask how well the organization knows its code.

Lawyers can best protect and advise their clients in all pertinent open source legal issues by obtaining or updating their certification in this area of their practice. Register for the free Black Duck Legal Specialist Certification Course and learn the fundamentals of open source while getting your continuing education credits.

SYNOPSYS®

The Synopsys difference

Synopsys provides integrated solutions that transform the way you build and deliver software, accelerating innovation while addressing business risk. With Synopsys, your developers can secure code as fast as they write it. Your development and DevSecOps teams can automate testing within development pipelines without compromising velocity. And your security teams can proactively manage risk and focus remediation efforts on what matters most to your organization. Our unmatched expertise helps you plan and execute any security initiative. Only Synopsys offers everything you need to build trust in your software.

For more information, go to www.synopsys.com/software.