Open source practices to support in-house innovation

Open source dominates the software industry and, by its very nature, appears to be at odds with IP protection. As open source packages continue to grow in popularity, in-house counsel should be aware of the pitfalls of their use and the complex IP landscape.

By Michael Moore, Judy Shie, Joe Kucera, Tarisa Wain and Ron Karr

Open source software usage and the associated licensing issues are one of the most common concerns facing technology IP attorneys today. This is particularly true now that innovation is performed more in software and less in dedicated hardware, such as semiconductor chip or programmable devices (eg, field-programmable gate arrays).

One of the primary benefits of software is the speed at which it can be developed, prototyped and iteratively tested, when compared to silicon or hardware solutions. One factor that enables this speed is the wide availability of public, pre-tested and readily available software (so-called ‘open source packages’ or simply ‘packages’ in software terminology) on repositories such as GitHub and SourceForge, among many others.

Open source packages are generally available to download by any member of the public, without requiring monetary payment. However, they come with an associated non-negotiable software copyright licence, which governs their use (sometimes commercially), distribution and modification. These open source software licences – particularly the so-called ‘copyleft’ or ‘viral’ flavours – use the legal copyright in the software as a mechanism to enforce certain behavioural norms within the broader software community.

The primary differences between open source software and closed source (proprietary) software relate to the payment of money and access to the underlying source code. With open source software, there is no warranty and typically no payment required. In addition, the express intent of the author and any subsequent contributor is that the source code be kept publicly available indefinitely, with clear and public attributions to the author or copyright holder. This is to allow the public to view, contribute to, improve, modify and use the software, within the bounds of the open source licence. Any documentation or user guides, to the extent that they exist, tend to be written for a technical audience and are less readily usable by a lay user.

Typically, no support is provided with open source packages – although in some cases, technical support may be purchased separately or provided under a commercial licence for the same software. Some open source packages are offered as part of a dual-licensing arrangement where the package is distributed under an open source licence to promote its adoption in the community; however, where commercial use or use with support or consulting services is desired, a separate commercial licence may be purchased from the copyright holder.

In contrast, with proprietary software (eg, a popular suite of office productivity tools), the source code is unavailable, copying is forbidden without purchase of a licence key and support is provided by the software vendor.

Issues for in-house counsel

There are certain concerns that practitioners – particularly in-house counsel at software companies – should consider around permitting or even encouraging use of open source packages.

Open source software is a useful tool and, when used correctly, can significantly improve productivity and speed to market for software products. With careful handling, it is a boon to advancement and product development; however, if mishandled, the results can be negative.

In the context of a typical fast-moving software company, developers (software engineers or programmers) are often under release deadlines to complete product testing and delivery as quickly as possible. This can lead to less careful developers simply ‘grabbing something’ off the Internet and incorporating it into the product for the sake of expediency.

Without careful and deliberate documentation of where the code was sourced from and the version of any package used, the company faces significant risks. These include copyright breach (legal consequences), functional deficiencies in the code (technical consequences) and even security vulnerabilities (data breach consequences) being introduced into the software, often through unpatched weaknesses in a stale version of an open source package.

In the context of intellectual property, developers may put the company’s assets at risk if open source software is not used correctly. For example, if developers are using open source in such a manner as to cause the viral licence to attach to company proprietary code, then the company may be obliged to make a portion of its proprietary source code public. If the company considers its proprietary source code to be a trade secret, this can be effectively lost, along with a portion of copyright value in the affected code. Alternately, a company may need to remediate the situation by spending significant resources rewriting components that have grown reliant on an inappropriately licensed open source component.
Interaction between patents and open source software

In the patent space, certain open source licences include a patent grant as a condition of the licence. The Apache 2.0 licence (www.apache.org/licenses/LICENSE-2.0) is one of the more popular licences in the industry and one which most commercial entities find palatable for use in products. The Apache 2.0 licence includes the following term:

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by its Contribution(s) alone or by combination of its Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

The Apache Foundation provides a thorough overview of the implications and obligations of this patent term in its FAQ (http://www.apache.org/foundation/license-faq.html).

The widely used general public licence (GPL) 2.0 v3 (https://www.gnu.org/licenses/gpl.html) also has a strong patent licensing term; however, the specific text is long (see Part 11 of the GPL), so will not be repeated here.

In-house counsel should bear in mind that if the developer team is using open source software in such a manner that the copyleft (viral) nature of the licence (eg, GPL) attaches to company proprietary code, then patents that the company holds relating to the code may become encumbered by the patent licence terms of the GPL.

Further, if the developers are contributing code into open source projects covered by such a patent-licensing provision, then any patents that the company holds which are necessarily infringed by the contribution to the open source project can be effectively encumbered by the open source commitment – and their value diminished accordingly.

In a practical context, it is important to train developers (particularly junior or less experienced ones) to obtain managerial and legal approval before pushing code out to external open source repositories and before pulling in open source code, particularly those with copyleft (viral) licensing terms.

Overlap and perceived conflict between open source and patents

The GPL 2.0 v3 has an interesting philosophical statement in the preamble:

“Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the
Practical tips: details to include in an intake review form

- Is the open source package intended for internal use only, or for external distribution?
  - If external, what distribution method is envisaged?
- What ability is there to update the code post-shipment?
- Can a fix be pushed out easily or is the code in a physical product (ie, isolated in a hardware appliance)?
- Correctly and clearly state any linkages used in the product.
- Provide a link to the source repository (eg, on GitHub) and suggest possible alternative packages with less onerous (ie, less copyleft) licences that would satisfy the same function.

The Linux Foundation provides a helpful set of suggestions: www2.thelinuxfoundation.org/l/6342/2012-03-28/7wb5k/6342/48826/lf_complicate_foss_approval_req.pdf

Practical tips: selecting a code-scanning tool

- Consider why you are scanning:
  - Is it for licence compliance and documenting attributions, to identify and remediate possible security vulnerabilities, or both?
  - Certain commercial tools are better at each of these functions, so pick a tool to best address your goal.
  - Where possible, combine licence compliance scans with security (common vulnerabilities and exposures) auditing.
- Look at the frequency of updates to the knowledge base on which the tool depends.
- Ask whether the tool is cloud-based or locally hosted. If it is local, who manages and resources it? Is there a commitment from the engineering team to support it and provide resources (eg, a virtual machine) for the tool to reside on?
- Verify the granularity of the scan provided by the tool. Does the tool scan package by package or down to individual code snippets within a package?
- This difference in granularity can lead to different scan outcomes.
- Ask whether the tool can automatically identify the package, version or multiple licences.
- Check how easy the scan tool is to operate and understand. Is it a ‘techie only’ tool or can non-technical users (eg, a lawyer) understand the output?
- Check whether the tool can be incorporated into your build environment to automatically scan as part of the code production or testing process.
- Check whether the tool can be used for M&A scanning of third-party external codes, as well as for production scanning of internal code. Can the tool support separate non-overlapping scan directories in order to avoid contamination of the internal production code with external third-party code?
- Check whether the tool can easily generate attribution documentation or a bill of materials. This can be a significant time-saver.

special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The broader philosophy of the GPL – and the (stereotypical) view of some software developers who favour the GPL – is that software patents are bad. The logical corollary to this is the philosophy held by some traditional IP attorneys: “Open source? No way!” As with many issues, the truth lies somewhere in the middle.

The pro-GPL software developer view stems from a perceived threat that patents pose to the open source community, as well as a distaste among software developers on the perceived evils of patent assertion entities (PAEs) and the desire not to generate patents that may fall into the hands of PAEs. This is a strongly held belief that in-house counsel must often overcome when developing patents in the context of software-centric organisations.

One analogy that may help to explain the concept to software developers is that patents and open source are often two sides of the same coin (innovation). Patents protect innovation (concepts and details of a technical solution), while open source protects speech (creative expression and a specific code implementation of the technical solution). Patents protect the concept, which may be implemented in multiple ways in a range of software languages. Copyright covers the written expression of one specific implementation typically using one specific software language (eg, code written in C++).

Open source software protects – via copyright – the freedom of expression to copy, distribute, modify, comment on and improve on a specific implementation of the code, by preventing the code from being locked down as a trade secret. Open source also helps to identify bugs and weaknesses in the code by permitting many eyes to peer review the code in parallel and thus provide different perspectives. In a famous law attributed to Linus Torvalds, the creator of Linux, “given enough eyeballs, all bugs are shallow” – meaning that, with sufficient peer review, functional flaws and security vulnerabilities are more likely to be discovered.

As mentioned, patents protect the underlying concepts, techniques or methods which are implemented in the solution – regardless of the software language used to code it. Patents preserve the ability of the creator (often a developer) to protect their innovation, while allowing the knowledge (concepts and details as expressed in the patent text or drawings) to become public for others to learn from, modify and improve on. In addition, if the code is released under an open source licence, the inventor or author can allow the
There are a wide variety of open source licences available, ranging from mild to wild. Some licences (eg, Massachusetts Institute of Technology (MIT), Berkeley Software Distribution (BSD) and Apache 2.0, as well as simple attribution-type licences) are considered fine and appropriate for commercial use. Others (eg, the Affero general public licence (AGPL)) are sometimes considered unsuitable for commercial use. There is a whole spectrum in between, with varying levels of obligation for the user and associated risk. One primary risk that in-house counsel should consider is the potential for an extension of obligations to company proprietary code caused by the use of open source licensed code alongside the proprietary code. This is referred to as the ’viral’ nature of certain open source licences, such as the GPL and the lesser GPL.

In 2015 popular open source repository site GitHub published Table 1, showing the approximate proportion of licence usage on the site. Over the past two years, the relative proportion of these licences has changed slightly, with MIT becoming more popular, according to certain commercial vendors.

### TABLE 1. Approximate proportion of licence usage

<table>
<thead>
<tr>
<th>Rank</th>
<th>Licence</th>
<th>% of projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MIT</td>
<td>44.69%</td>
</tr>
<tr>
<td>2</td>
<td>Other</td>
<td>15.68%</td>
</tr>
<tr>
<td>3</td>
<td>GPLv2</td>
<td>12.96%</td>
</tr>
<tr>
<td>4</td>
<td>Apache</td>
<td>11.19%</td>
</tr>
<tr>
<td>5</td>
<td>GPLv3</td>
<td>8.88%</td>
</tr>
<tr>
<td>6</td>
<td>BSD 3-clause</td>
<td>4.53%</td>
</tr>
<tr>
<td>7</td>
<td>Unlicense</td>
<td>1.87%</td>
</tr>
<tr>
<td>8</td>
<td>BSD 2-clause</td>
<td>1.7%</td>
</tr>
<tr>
<td>9</td>
<td>LGPLv3</td>
<td>1.3%</td>
</tr>
<tr>
<td>10</td>
<td>AGPLv3 (Affero)</td>
<td>1.05%</td>
</tr>
</tbody>
</table>

Source: https://blog.github.com/2015-03-09-open-source-license-usage-on-github-com/
Train your developers to understand and respect it, as well as to engage the legal team early in the process, and everyone will benefit from improved productivity.

Know your open source licences – understand which licences may have copyleft or viral effects on proprietary code and which licences are permissive.

Train developers about the risks associated with open source licences, including the potential risk of encumbering company patents.

Work with the developer team to track and review all open source software usage.

Be aware of, and sensitive to, the time constraints on the developer team – scan your code well in advance of the release date and have a plan to deal with any problems raised.

Maintain accurate documentation of open source software used in each software product.

**Open source software in M&A**

In the context of M&A, open source diligence is an important and often complex process. The acquirer wants to understand any pre-existing obligations on the codebase and potential encumbrances on any patents that may be part of the target company. These obligations and encumbrances may significantly affect the deal value, as the acquirer must avoid purchasing a problem if there are hidden obligations in the target company. Similarly, the target company must understand and be able to demonstrate clear compliance with any such open source obligations in order to improve the likelihood of a successful sale and maximise the value received.

The target company should also endeavour to show a complete and clear listing of all open source packages used in its product(s), along with version information, distribution status and whether commercial licences have been purchased for certain packages. Where the target company demonstrates good discipline and clear internal processes and policies around selection and approval of open source licences, as well as the packages that it uses, this can go a long way to reducing concerns on the part of the acquirer.

It is still incumbent on the acquirer to perform thorough due diligence on the codebase before the acquisition. One of the most effective ways to do this is to have an objective third party scan the entire codebase using a commercial scanning tool. There are several of these tools and service providers available, and the acquirer should familiarise itself with the quality, reputation and price of services of the various commercial vendors. A service provider should be engaged well ahead of any prospective acquisition in order to avoid last-minute surge pricing during a deal. The scanning service provider market is competitive, so acquirers will likely find the best pricing by shopping around and comparing quotes from multiple vendors.

On receiving a scan report from the service provider, the acquirer should carefully compare this to any self-reported open source listing from the target company. Any significant discrepancy should be a red flag to the acquirer that the target’s internal diligence process may be weak and may need further training and careful oversight. If the scan report flags the use of packages with copyleft licences in a manner that creates obligations for the target company to disclose proprietary code, or which encumbers valued patents of the target company, this should be considered as part of the overall deal value. An acquirer may consider requesting the target company to (where possible under a dual-licence scheme) obtain commercial licences to the problematic open source packages. Alternatively, the acquirer may have to factor in a cost and timeline to rip and replace the problematic open source packages in order to bring the target company into compliance with its licensing obligations, affecting the deal value and attractiveness accordingly.

> “Patents and open source are often two sides of the same coin (innovation). Patents protect innovation, while open source protects speech”

**Tracking open source software bill of materials and changes**

An accurate bill of materials should be compiled and maintained by the engineering or development team for every version of the software product released. Just as with a physical product, a bill of materials identifies the open source packages and versions used in creating the software product – this is essential documentation for handling future support issues. Any modifications made to the open source packages should be documented in the source code, either through comments in the modified file or in a separate comments file. If the developer wants their modifications propagated into future versions of the open source software, they may consider contributing their modifications back to the open source software project, after first confirming with the legal department that such a contribution would not negatively affect the company’s IP rights.

**Action plan**

Like fire, open source is an effective and valuable tool when used appropriately.

- Train your developers to understand and respect it, as well as to engage the legal team early in the process, and everyone will benefit from improved productivity.
- Know your open source licences – understand which licences may have copyleft or viral effects on proprietary code and which licences are permissive.
- Train developers about the risks associated with open source licences, including the potential risk of encumbering company patents.
- Work with the developer team to track and review all open source software usage.
- Be aware of, and sensitive to, the time constraints on the developer team – scan your code well in advance of the release date and have a plan to deal with any problems raised.
- Maintain accurate documentation of open source software used in each software product.

---

**Michael Moore** is vice president and assistant general counsel of products and intellectual property, **Judy Shie** is product legal counsel, **Joe Kucera** is senior IP strategy manager, **Tarisa Wain** is product and patent lead programme manager and **Ron Karr** is director of technical strategy, chief technology officer’s office at Pure Storage Inc.