Open Source Compliance and Goldilocks: Too Little, Too Much, Just Right

Robert Marion [(a)]

(a) Sr. Open Source Compliance Analyst at Citrix

DOI: 10.5033/jolts.v10i1.138

Abstract
Although open source compliance can be tricky and require resources it is not inherently necessary to employ dedicated personnel and enterprise solutions to accomplish a reasonable degree of best practices. This is particularly in situations where any resource allocation will compete for the resources required to actually create a product.†

Keywords
Law; information technology; Free and Open Source Software; licensing; compliance

Software developers depend on open source code. They pull it from GitHub, SourceForge, the Python Package Index (PyPI) and many other sources. There are Open Source advocates, who believe all software should be open source and there are professional organizations like OpenChain[1], the Open Source Initiative[2] and the Free Software Foundation[3] which assist in defining open source compliance programs. There are also foundations that produce, host and represent the Open Source community such as the Apache Software Foundation.

In contrast, businesses often substantially depend on their proprietary code to build products and generate revenue. They have a need to demonstrate that they know what is in their code because a software product is an assembly of many different components and these components may be open source projects (or commercial) and will be comprised of multiple authors, copyrights, and licenses. The many parts of a software product are often obscured by the ease with which one can incorporate code that does not originate with the product's developers. Entire software architectures can be built around a third-party framework before consideration is given to whether its license is compatible with proprietary code. One example is that in the case of the General Public License (GPL), other open source licenses may be non-compatible with the GPL's terms. A license establishes both rights and responsibilities and a failure to understand the responsibilities of a license can lead to litigation, monetary losses, missed deadlines and the need to halt shipment of a product. A growing awareness of licensing obligations with the use of third-party open source means that many companies now have introduced mechanisms for compliance. However, with a startup company, the compliance

† The views and opinions expressed in this article are those of the author and do not necessarily reflect the policies, opinions or position of Citrix Systems.
1 https://www.openchainproject.org
2 https://opensource.org/
3 https://www.fsf.org/

program may not exist at all or it may be as light as asking the developers to be wary of using code that is "adversely" licensed. Even with larger companies there may only be a single person responsible, often a lawyer, or a department of persons using enterprise software tools to prevent the inclusion or detect existing code that has licenses that are not considered proprietary friendly.

This article will consider the notion of there being a waterline for a right amount of compliance. A tech startup will not, nor should they have a multi-person compliance department any more than it would make sense for a company with hundreds or thousands of developers to have a single person who may have other duties running a compliance program. In all cases, there is a need to balance licensing compliance with the amount of resources available to a business.

No Free Beer for You

How Free Open Source Software (FOSS) is defined can be somewhat contentious with various organizations evangelizing their perspective of what is reasonable and correct. Some entities and groups share a belief that software should be freely shared for the benefit of all humankind, or at least, all programmer-kind. It is an altruistic philosophy, a democratic ideal that software products should be thought of as ideas that do not belong to a single class of people. The Free Software Foundation defines free open source software as having the freedoms to use, run, modify and redistribute a program. We "should think of free as in free speech, not as in free beer".[4] This belief is often at odds with the imperatives of the business world, which seeks to monetize everything possible to provide returns to their stakeholders. Yet somewhat ironically, commercial enterprises benefit most from open source.

Open Source licenses mostly fall into one of the following buckets: Public Domain (not actually a license, but a dedication), Attribution Style (such as the MIT or BSD), weak copyleft (LGPL), and strong copyleft (GPL or AGPL). When it comes to compliance, the copyleft licenses are less "proprietary friendly" than the others and some compliance programs only focus on those license types via various degrees of exclusion. Increasingly, companies are concerning themselves with the greater task of finding all the open source in their products and creating a Bill of Materials. An example is that your mobile phone – whether from Apple or an Android provider - will contain a list of the open source third party software present. Companies are also placing their catalogues of open source components online. For example, you can see all the third party open source that goes into Ford Motor Company's infotainment system at https://corporate.ford.com/legal/ford-open-source.html. It is an extensive list. By creating this information and placing it online, Ford is telling its customers that it is compliant and that it has a process in place. That process is designed to instil confidence on the part of Ford vendors and customers that they themselves will know what is in their products (if, for example, they were to incorporate Ford software). Of equal importance is the capability of knowing whether a third party component contains software vulnerabilities. The famous Equifax hack resulted in millions of credentials being lost, costly legal repercussions, and the firing of CEO Rick Smith. Could this particular breach have been circumvented? Yes. Equifax was using a third party component with a known and published vulnerability. Not knowing what they had in their software cost them millions of dollars and a seriously tarnished reputation.

Why should anyone care? After all, when I buy/license software I am trusting someone else to take care of compliance. So Ford, or the Acme Anvil Company or any other enterprise from which I purchase (actually, license) software is responsible for knowing what they have provided and what their legal obligations may be. Yet unless my engagement with the software involves no further distribution this stance does not hold true, as copyright licenses such as those found in software apply on any instance of distribution. If you inherit a problem, it can become your problem if adequate understanding is not present. Invariably, whenever a licensing snafu becomes public, the responsible

4 https://www.gnu.org/philosophy/free-sw.en.html

company will blame an outside consultant, contractor … whomever, but this stance in no way mitigates their responsibility to make sure what they inherited and subsequently passed on through the supply chain is correctly and sufficiently compliant. Hence the need for a Bill of Materials (BoM).

Compliance must address one issue: licensing obligations must be met. If your goal is to conform with licensing terms, you must give proper attribution where it is required. The BSD and MIT are examples of attribution style licenses. If a license is copyleft, then the user's obligations are greatly increased and so, by the way, is the likelihood of litigation. It is my personal contention that fear of litigation should not be the motivation behind meeting one's licensing obligations. There are two better reasons. First, obeying FOSS licensing terms is the right thing to do from an ethical perspective. The Zlib license expresses this in plain English: "you must not claim that you wrote the original software." That should be easy enough to understand. Additionally, companies and persons who are reputable should have an easier time selling their products if they are perceived as being fair and ethical and software-community friendly. As mentioned, the ability to produce a BoM may be a necessity for management, and it equally can be simply appreciated by customers as an example of care and consideration for both acknowledged authorship and legal correctness.

Why is Compliance Difficult?

Both the proliferation of open source licenses and terms of certain licenses have made compliance a non-trivial matter. As of this writing, there are over eighty open source licenses listed by the Open Source Initiative. For a quick comparison of many of these licenses in table format, see the Wikipedia article 'Comparison of free and open-source software licenses."[5]

The GPL v2.0 license has 2,965 words. The GPL v3.0 clocks in at a hefty 5,660 words. Copyleft licenses evoke ideas such as derivative works, methods of compilation, distribution, aggregate works, etc. Some licenses that are commonly applied to software appear to be designed for other works of creative good such as the Creative Commons family of licenses and their origin in literature, imagery and similar works. Sometimes, knowing what licensing terms are applicable and how their obligations can be met is a sticky situation. Software engineers are not expected to be lawyers or licensing experts – but they do need to be familiar with the basic concepts of licensing. Furthermore, it is common to see upward of twenty different types of FOSS licenses in a single product. One product I recently looked at had over fifty different licenses or license combinations. Those are a lot of terms with which one must be familiar! This is not a simple matter if the correct approach is not used.

Complicating the issue of compliance is that software engineers can be notoriously independent minded given the nature of their work. Even if they are aware of licensing issues, they cannot be inherently trusted to be diligent given the practical difficulties imposed by writing software that works and delivering it on time. They may simply have other priorities, a situation that can be understood without judgement, but with the observation that it exists in production environments.

With all that said, one of the most difficult challenges with meeting FOSS obligations is knowing what is in your code. Not too long ago, it would be possible to pull the code from a repository and analyse it for third party materials. This has become much more challenging with the growing popularity of build systems that assemble code at build time (or even run time). For example, why have a hard copy of jQuery in your company's repository if you can fetch the most up-to-date version of what you need from an online CDN (Content Delivery Network)? Software as a Service such as The Cloud, and containers such as Docker, make knowing what goes into a build even more

5 https://en.wikipedia.org/wiki/Comparison_of_free_and_open-source_software_licenses

difficult. As alluded above, modern build processes may define what code will go into a program, but that code does not necessarily need to reside on your servers. This is becoming the norm rather than the exception and it creates a further challenge in knowing what third party open source you are building into your product.

Psst … You Wanna Buy a Company?

Mergers and acquisitions (M&A's) are an important part of the tech ecosphere. When larger technology companies wish to acquire another company, both sides must perform their due diligence. A business that is acquiring a software company will want to know that the software assets they are purchasing can be monetized. A company that is being bought may need to prove that the software they built has commercial value. The target company may need to prove that the intellectual property they claim is theirs actually belongs to them or is freely available under a proprietary-friendly license. If the software assets are unknown (the company cannot identify what goes into their build) or is built on non-proprietary-friendly licensed software, then it is possible for a deal to fail or the acquisition terms may be re-evaluated. Neither option represents a favourable outcome with the exceptional of marginal benefits for the purchaser in revising acquisition costs down.

It follows that an acquiring company will almost always seek the assistance of a law firm that specializes in these types of deals. When it comes to inspecting code for licensing issues … well, there are companies that specialize in that also.

What is the Right Amount of Compliance

The right amount of compliance not only depends on the size of the company, but the type of software product you are developing and how it is distributed. Projects may be entirely open source, may be proprietary but have open source APIs or may be what is now referred to as open-core. Consumer electronics and mobile apps are a much higher risk than web services because of the distribution model of each. That is to say, in the cases of web services, it is likely that third party components sitting in the cloud are often not considered to be distributed with the exception of code under the Affero GPL, which provides some mechanisms to address this use-case.

Open-core is a new name for something that has been going on for years. It is a combination of open and closed source, with the chief idea being that you can get some community contribution and acceptance and allow for some usage of a product without such users needing to purchase the software except in certain situations. These situations can often be defined as the full enterprise experience, various additions which will not in themselves be free or open source. This approach appears to be achieving a greater degree of popularity than it once had and, now that it has a name, also seems to have gained greater legitimacy. From an open source compliance perspective, open-core can be difficult to manage, and the cut-off point between the open code and the closed additions can cause friction.

The Cloud has changed what it means to deliver software. Not too long ago, purchasing software meant getting a media disk or downloading a product onto your server. That model of software delivery is becoming less relevant given how software products are delivered today, with The Cloud or SaaS rising dramatically and on-premise solutions are becoming less popular. This is important to open source licensing because most licenses were written before cloud computing existed. It is legitimate to believe that a GPL licensed software library may be used in the cloud and that does not constitute delivery and therefore the terms of the GPL license are not relevant to that situation. The Affero GPL (AGPL) license addresses that situation, but it has not been widely adopted, and certainly has not found favor with any significant number of companies in this space.

Let's pull back to the assertion from earlier that open source compliance is not easy. Years ago, when I was developing software for a large company, one of my colleagues handed me a clipboard and told me that, if I ever used any GPLd software, I should log it on the clipboard. I nodded my head knowingly and he left. I had no idea what the GPL was and there was little danger of anything being recorded on that clipboard. Fortunately, as I recall, I wrote almost everything by hand. Today, there is a word for people who write everything by hand: unemployed.

So how much compliance is required? That is sort of a trick question. If you and your company are developing software that will be distributed, then you and your company are expected to obey all intellectual property laws and be aware of the licenses of all third party materials incorporated into your software. In a very real sense, I have never seen 100% compliance and mostly there are no consequences. However, M&As have been aborted when it was discovered through an IP audit that the chief product being acquired was really a derivative work of GPL licensed software. One company, for which I was contracting, had been sued in the past for not complying with the GPL and I have seen one case where a company purchased a competing company's product so that it could analyze it to determine if they were violating any software licenses (especially the GPL). I call this "weaponizing" the GPL. These things do not happen often, but they do happen. For example, former Linux developer Patrick McHardy sued Geniatech and over thirty other companies in German courts over purported violations of the GPL. McHardy was unsuccessful in the Geniatech case, but his efforts overwhelmingly appear to be financial gain and not the safe-guarding of open source principles.[6]

Compliance is a Process

Some companies take a "ticking the box" approach. They may purchase some expensive software that promises to analyse their code and produce a report of all the third party components along with their licenses. I have evaluated many of these products and none of them can really accomplish the goal of sufficiently discovering third party materials. Why? There are a variety of reasons. Much of it has to do with how modern software is built. It is (or was) common for jQuery to be pulled in at run time by grabbing code from a "trusted" repo when it is needed. Sometimes code is pulled in programmatically at run time. Another example is Docker. Docker provides a platform-as-a-service and it also complicates life. Source code can be wholly taken from a third party project and it can even retain attribution, but it may be undetectable by enterprise scan tools. Such tools tend to do a great job at detecting unmodified binaries because they have a unique hash signature that can identify them. However, a simple JavaScript code section or snippet may go undetected.

It follows that simply purchasing a product and running scans is unlikely to produce a desired level of compliance and that a coherent process is instead required. From what I have seen, OpenChain has done an incredible job of defining those process inflection points, policy needs, and training approaches. For small companies, the overhead, time and effort required to implement those processes may be unaffordable in terms of both time and money. Instead, they may wish to do the following: train their engineers in open source licensing concepts. This may consist of a brief course given online or in-person. A trained engineer will know, most importantly, to record the license of a third party component. They may seek permission informally or formally and make an intelligent decision as to whether a given library poses an intellectual property risk. A small company that produces a non-trivial amount of code may also want to have an annual IP checkup by a consulting company.

Whether a company is large or small, the first step to compliance is training. A well-trained engineering staff, including managers, will ask the important questions before they become problems. Catching a problem after third party materials are introduced into your codebase may

6 https://www.zdnet.com/article/linux-beats-internal-legal-threat/

require costly refactoring and testing. So, the first step, whether a company is small or large, is to educate all involved in the production of a software product in basic license compliance. The Linux Foundation has a great course[7] that is accessible to beginners and also useful to those who are already familiar with FOSS. I also highly recommend the following video by IP lawyer Heather Meeker.[8] If training is not mandatory, then it may not be effective. Training is the very foundation of compliance.

Formal documented procedures, policies, and resources must be a part every company's FOSS compliance program. Policies should include more than how engineers should handle copyleft type licenses, but attribution style licenses, commercial licenses, situations where a license is unknown or does not exist. Policies must also consider inbound and outbound materials. Of equal importance is knowing what the workflow should be when licensed material, which is incompatible with a company's intellectual property goals, is found in their codebase. Employees must know where they can get answers regarding the use of a particular license, what to do if something appears to be unlicensed, what to do if licensing terms are complex, whether they are allowed to contribute to outside open source projects and so on. Knowing where these documents are stored can be part of the training program. Really well written, thoughtful documentation is only helpful if it is easily accessible. For some no cost policy and procedures suggestions, I recommend visiting the Blue Oak Council website. [9]

Producing a Bill of Materials for your products demonstrates to your own staff and your customers that you know what is in your code and that your company is serious about diligence in intellectual property and licensing matters. You cannot fix a problem if you do not know that it exists. For small codebases, this may be a manual effort. Larger codebases will require some type of discovery tool. Naturally, the larger the codebase, the greater the effort will be required to produce a bill of materials. A variety of tools exist to help with this problem, and they can be expensive, complicated enterprise tools, or they themselves may be open source. Smaller companies will naturally gravitate toward simpler, less expensive solutions or they may outsource the effort. Outsourcing tends not to scale up in terms of cost, so larger companies are more likely to have in-house staff. There are standards for Bill of Materials around open source, with SPDX being a key example with growing adoption for both manual and automated review processes[10].

Open source audits are not only crucial for M&A activity but should be incorporated into the software lifecycle. It can be understood that for smaller companies an M&A may be the only time an audit occurs, and it should equally be understood that it pays to be wary as such deals have failed due to a lack of diligence. Larger companies will want to keep continual tabs on their licensing exposure. Some companies are the proverbial Red Shirts of Star Trek fame. That is to say, they have a target on their back and parties aiming at that target may be hackers, customers, or even their competition. Although FOSS lawsuits are not very common, there have been occasions where licensing has been "weaponized" as in the case of CoKinetic Systems Corp. vs Panasonic Avionics.[11] Audits can be performed in-house, automated in the pipeline, or outsourced. There are an increasing number of tools, some free and some quite expensive, that can be employed for this effort. It is important to study the available tools and choose the appropriate ones given the time and money will be invested in implementing them. One of the best investments a company can make is choosing the right personnel to implement such programs, often those with a technical and legal background, who may have contributed to open source projects or be active in open source advocacy organizations.

7 https://training.linuxfoundation.org/training/beginner-guide-to-oss-development-lfd102/
8 https://www.youtube.com/watch?v=gF4b1TA5Q5w&list=PLAVikl6VpxPeBtplWOnfzNmiUz529AYAy
9 https://blueoakcouncil.org/
10 https://spdx.org
11 https://resources.whitesourcesoftware.com/blog-whitesource/the-100-million-case-for-open-source-license-compliance

Conclusion

Although open source compliance can be tricky and require resources it is not inherently necessary to employ dedicated personnel and enterprise solutions to accomplish a reasonable degree of best practices. This is particularly in situations where any resource allocation will compete for the resources required to actually create a product. This situation will often be the case with smaller companies and that is understandable that choices need to be made. However, compliance training courses, policy examples, and extensive information about effective processes can be found online. The first time a customer requests a Bill of Materials may be the catalyst for building such a program but it is hardly the sole reason for having one or regarding such activity as time limited. As a company grows, the compliance function should also grow to ensure ongoing effectiveness in the use of third party copyright for protection, for effectiveness and for solid positioning in both product and M&A activities. Over time and as experience grows the compliance function tends to move from legal review to operations and engineering. This saves the lawyers for the more difficult or unusual cases and reflects the fact that mature companies are expected to have trained personnel and processes in place. FOSS compliance is no longer an option – it is a necessity – and today it is a necessity that can be accomplished by organizations of any size and at any stage of growth with a little effort.

Further Reading

https://techcrunch.com/2019/01/12/how-open-source-software-took-over-the-world/

https://corporate.ford.com/legal/ford-open-source.html

https://www.zdnet.com/article/cern-leaves-microsoft-programs-behind-for-open-source-software/

https://arstechnica.com/information-technology/2017/09/massive-equifax-breach-caused-by-failure-to-patch-two-month-old-bug/

About the author

Robert Marion *is an Open Source Analyst and software engineer who has focused on open source compliance for over ten years and who has trained numerous corporations in multiple countries in open source compliance and compliance enterprise tools. Mr. Marion still writes software. When he's not working with code he enjoys playing guitar and touring the country on his motorcycle.*