KILPATRICK TOWNSEND

A Practical Guide to Open Source Software[1]

Michael Pavento

I. Introduction

Open source software ("OSS") is software that is freely available in source-code form for anyone to use, copy, modify and distribute. Generally speaking, however, OSS is not "public domain" software. Like any other software, OSS is copyrighted intellectual property. Authors have the right to control or condition the use of their original OSS code and typically do so through license agreements. Unlike traditional software licenses that seek to limit or prohibit further dissemination of the licensed software and certainly the underlying source code, OSS license agreements generally seek to encourage dissemination and to ensure that the source code remains open and accessible to all. Like any other software acquired from an outside source, the applicable license agreement is the starting point for understanding and managing the obligations imposed upon and the risks undertaken by an organization with respect to OSS.

A careful review of some of the more common license agreements, for example the *GNU General Public Licenses*, the *Apache License*, the *Common Development and Distribution License* and the *Berkley Software Development License*,[2] will dispel some of the common misperceptions regarding OSS. For example, it is sometimes said that OSS cannot be used with proprietary software. To the contrary, OSS licenses do not limit the manner in which the licensed OSS may be used; they typically permit all uses, but impose reciprocal obligations that are triggered when the licensee distributes the licensed OSS or modifications of it. It is also often said that all OSS licenses require the release of source code for any proprietary software used in conjunction with the OSS. While some OSS licenses impose on the licensee an obligation to publish source code for works combined with or otherwise derived from the licensed OSS, it is certainly not the case that *all* uses of the licensed OSS in conjunction with proprietary software result in derivative works, or that all OSS licenses extend to derivative works.

OSS is pervasive in today's software development environments. The notion of adopting a corporate policy prohibiting the use of all OSS is generally not a workable solution. Companies that acquire or develop software should strongly consider adopting a well-articulated policy regarding the use and management of OSS within their organizations, so that they can be aware of their rights and obligations under applicable OSS license agreements and implement reasonable procedures to ensure compliance.

II. OSS Licenses

Open source license agreements have been created and disseminated by nonprofit software foundations and consortiums (*e.g.*, Free Software Foundation, Apache Software Foundation, World Wide Web Consortium), universities (*e.g.*, University of California, Berkley, University of Illinois, Massachusetts Institute of Technology), for-profit corporations (*e.g.*, Microsoft, Apple, IBM, Oracle), government agencies (*e.g.*, NASA) and individuals. A list of common OSS license agreements is published by the Open Source Initiative.[3] The Open Source Initiative is a nonprofit organization that, among other things, promulgates its widely accepted "Open Source Definition"[4] and approves OSS license agreements that it deems to be in compliance with that definition. The originator of a new

[1] This article is adapted from Michael Pavento, *Open Source Software: A Practical Guide for In-House Counsel*, in Association of Corporate Counsel (September 2012).
[2] Open Source Initiative, Licenses by Name (June 24, 2013, 4:30pm), http://opensource.org/licenses/alphabetical.
[3] Open Source Licenses (June 24, 2013, 4:30pm), http://opensource.org/licenses/index.html.
[4] *See* http://opensource.org/docs/osd.

OSS project typically elects to release the OSS under one of the approved OSS license agreements, but may also choose to use a customized version (when permitted) or even create a new license agreement.

The basic concept common to all OSS license agreements is that they seek to ensure that all downstream users have the freedom to use, modify and distribute the licensed OSS. More permissive OSS license agreements impose minimal obligations on the licensee, such as obligations to maintain attribution and legal notices. These agreements typically permit modifications to the OSS and allow such modifications to be distributed under any license, whether open source or proprietary, of the licensee's choosing. Some of these more permissive license agreements require the licensee to make the source code of the licensed OSS available to those who acquire the OSS from the licensee, though others do not include such a requirement. The more restrictive OSS licenses, often referred to as "copyleft" or "viral" license agreements, impose obligations on the licensee not only with respect to the licensed OSS but also with respect to any works derived from it.

It is important to note that virtually all OSS license agreements impose obligations and conditions on the licensee *only* when and if the licensee distributes the OSS and/or modifications of it.[5] Therefore, if a company acquires certain OSS and uses and/or modifies it solely for internal purposes, then the company will not be required to take any affirmative actions to comply with the applicable OSS license agreement. That is not to say, however, that internal use of OSS is free of risk. Most OSS license agreements state that the OSS is provided "as is," and disclaim any and all liabilities on behalf of all upstream contributors. This means that OSS is provided without warranty and, in most cases, without any expectation of formal support or maintenance services from the licensor – although informal support is often available from the open source community for the more popular OSS projects and commercial (*i.e.*, paid) support also exists for certain major OSS projects like JBoss, Linux and MySQL.

III. Derivative Works and Copyleft

One of the biggest fears associated with OSS is that its use in conjunction with proprietary software will result in a derivative work and that the source code of the derivative work must be made available free of charge to downstream recipients with broad permissions to modify and redistribute it. This could have significant implications for companies that distribute proprietary software and is clearly of concern for OSS governed by license agreements with "copyleft" provisions, such as the *GNU General Public Licenses*; the *Mozilla Public License, Version 2.0*; the *Common Development and Distribution License, Version 1.0;* and the *Eclipse Public License, Version 1.0*.[6] Understanding the concept of a derivate work in the context of an OSS license is therefore critical. Unfortunately, however, determining what is and what is not a derivative work in this context is sometimes a difficult exercise. To make matters worse, little case law exists from which to draw guidance.

Some OSS license agreements use the term "derivative work" without defining its intended meaning.[7] In these instances, the term is likely to have the meaning attributed to it under copyright law. Other OSS license agreements, such as the *GNU General Public Licenses,* refer specifically to the copyright law when defining terms intended to encompass derivative works. In particular, the *GNU General Public License, Version 2,* provides:

> … a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language…[8]

Similarly, the *GNU General Public License, Version 3,* provides:

> To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.[9]

When it comes to copyright law, the derivative work inquiry is highly fact-specific. United States copyright law defines a derivative work as:

> "… a work based upon one or more pre-existing works, such as… any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations,

[5] *But see* Section VI, Implications of OSS Licensing for Cloud Computing, *infra*.
[6] Open Source Licenses by Category, http://opensource.org/licenses/category.
[7] *See*, *e.g.*, Eclipse Public License, Version 1.0 ("'Contribution" means: … b) in the case of each subsequent Contributor: i) changes to the Program, and ii) additions to the Program… Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.").
[8] GNU General Public License, Version 2, section 0.
[9] GNU General Public License, Version 3, section 0.

elaborations, or other modifications which, as a whole, represent an original work of authorship, is a 'derivative work'."[10]

Generally speaking, if a modified work is deemed to infringe the copyright of the original work, then it is a derivative work of the original work.[11] A threshold question in the infringement analysis is whether there is "substantial similarity" between the two works. Substantial similarity between two works is typically found when the similarities between their copyrightable elements are more than *de minimis*. A finding of substantial similarity, which is a question of fact to be decided by a jury, may turn not only on the amount of copying but also whether the portion copied constitutes the "heart" of the original work.[12]

Relying on copyright law principles to determine whether derivative works result from certain uses of OSS thus requires a case-by-case analysis of the similarities between the OSS and the licensee's software. For example, in some instances it may be valid to argue that copying 15 lines of code into the licensee's software from an OSS file that includes 5,000 lines of code is not substantial. In other instances, copying only those 15 lines of code may be deemed to be copying the "heart" or central portion of the OSS code. Also, even if the licensee does not directly modify or copy from an OSS file, certain uses of it with other software may result in a combination that could be considered a derivative work. Therefore, when dealing with OSS licenses that rely on copyright law principles, a thorough investigation of how much and what part of the OSS code is copied or modified and/or how the OSS is used needs to be made in order to anticipate or predict how a court might rule on the derivative work issue.

Still other OSS license agreements provide their own definitions of the term "derivative works"[13] or related terms like "modification."[14] In contrast to the substantial copying inquiry under copyright law, these definitions of "derivative works" and the like tend to encompass any new file that contains *any part* of the licensed OSS. Thus, with these types of OSS licenses, copying even a snippet, however small, from the licensed OSS into a file with proprietary code will arguably result in a derivative work as defined by these types of agreements. Similarly, making any minor modifications to an OSS file could arguably be considered a derivative work under these rigorous definitions.

Even those who take the most aggressive positions with respect to derivative works seem to agree that mere aggregation of separate software components on the same medium (*e.g.*, in a computer memory, on an optical disc, etc.) does not result in a derivative work. Another seemingly easy case is where two software components are compiled into a single program, such as an executable program; most would agree that the resulting program is a derivative work. For many other cases, however, there is no simple answer. Many questions remain unanswered and opinions vary as to whether certain combinations of unmodified OSS with other software (*e.g.*, through linking, the use of plug-ins, or use of Javascript) result in derivative works.

A. The Linking Debate

Software libraries are files that include content like configuration data, subroutines, classes, values, or type specifications. Executable files (*e.g.*, application programs) make reference to libraries through a process known as linking. There are two types of linking: *static linking* and *dynamic linking*. Static linking is performed during the creation of an executable. In essence, a statically linked library is copied into the executable file. Dynamic linking, on the other hand, is performed when an executable is loaded into computer memory or when it is actually executed. In this case, the library remains a standalone file that is referenced "on-the-fly" by the executable; it is not copied into the executable.

[10] 17 U.S.C. § 101 (2013).
[11] *See* Melville B. Nimmer and David B. Nimmer, *Nimmer On Copyright* § 3.01, (Rev. Ed. 2013) ("a work will be considered a derivative work only if it would be considered an infringing work…").
[12] For a more detailed discussion of derivative works, particularly in the context of software, *see, e.g.*, Natalie Heineman, *Computer Software Derivative Works: The Calm before the Storm*, 8 J. HIGH TECH. L. 235 (2008); Lothar Determann, *Dangerous Liaisons—Software Combinations As Derivative Works? Distribution, Installation, And Execution Of Linked Programs Under Copyright Law, Commercial Licenses, And The GPL*, BERKELEY TECH. L. J. 4, 1421 (2006). *See* also *Micro Star v. Formgen*, 154 F.3d 1107, 1110 (9th Cir. 1998); *Lewis Galoob Toys, Inc. v. Nintendo of Am., Inc.*, 964 F.2d 965 (9th Cir. 1992), cert. denied, 507 U.S. 85 (1993). *See* generally Melville B. Nimmer, *Nimmer On Copyright*, §§ 3.03[A] and 6.06[B] (Rev. Ed. 2013).
[13] *See*, *e.g.*, Apache License, Version 2.0 ("'Derivative Works' shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.").
[14] *See*, *e.g.*, Mozilla Public License, Version 2.0 ("'Modifications' means any of the following: a. any file in Source Code Form that results from an addition to, deletion from, or modification of the contents of Covered Software; or b. any new file in Source Code Form that contains any Covered Software."); *See* also, *e.g.*, Common Development and Distribution License, Version 1.0 ("'Modifications' means the Source Code and Executable form of any of the following: A. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications; B. Any new file that contains any part of the Original Software or previous Modification; or C. Any new file that is contributed or otherwise made available under the terms of this License.").

It is difficult to refute the notion that statically linking an OSS library with a proprietary executable file results in a derivative work of the OSS library; it is basically equivalent to copying portions of the OSS code from the library file into the proprietary code. Dynamically linking an OSS library with a proprietary executable file, however, is more controversial. Some argue that dynamic linking does not result in a derivative work of the OSS library, because the two pieces of software are distributed as separate entities and are never permanently combined. Common counter-arguments are that the proprietary code will often include the header files or other interfaces of the dynamically linked files (*e.g.*, some portion of the OSS is in fact copied into the proprietary code) and that the linkage between the two files temporarily results in a combined work that includes both the proprietary code and the OSS. Some of the dynamic linking arguments regarding derivative works may be questionable under strict copyright principles (*e.g.*, the header files of the OSS may not be copyrightable in the first place). Nevertheless, it should be noted that dynamic linking is performed differently by different operating systems and for code written in different programming languages. Therefore, the derivative work analysis will depend on the particular details of each situation, which will likely require input from expert software developers.

The Free Software Foundation ("FSF"), which created the *GNU General Public Licenses* and holds the copyright to several popular OSS programs,[15] takes a very aggressive position with respect to dynamic linking and derivative works. On its website, the FSF states that "[l]inking [an OSS library] statically or dynamically with other modules is making a combined work based on [the OSS library]. Thus, the terms and conditions of the GNU General Public License cover the whole combination."[16] In the view of the FSF, this is true regardless of whether the OSS library is distributed together with the other modules or separately from the other modules and even if the OSS library and the other modules are distributed by different parties.[17] This view is not necessarily shared by all commentators,[18] but undoubtedly causes many companies and their legal advisors to take a conservative approach when it comes to the use of OSS governed by the *GNU General Public Licenses.*

B. Plug-Ins and Javascript

Similar to dynamically linked libraries, software plug-ins present issues with respect to derivative works under certain OSS license agreements. A plug-in is a software component (or a set of components) that is separate from and provides additional functionality to a software application program. Plug-ins such as Adobe Flash Player, QuickTime and Java applets are commonly used in web browsers to add video player functionality. Software applications must be configured with programing interfaces to support the use of various plug-ins, so the question becomes whether the combination of the software application and a plug-in results in a derivative work when either the application or the plug-in is governed by a copyleft OSS license.

The FSF takes the position that the manner in which the application program interacts with a plug-in determines whether a derivative work is created in the context of the *GNU General Public Licenses*. The FSF states on its website:

> It depends on how the program invokes its plug-ins. If the program uses fork and exec to invoke plug-ins, then the plug-ins are separate programs, so the license for the main program makes no requirements for them.
>
> If the program dynamically links plug-ins, and they make function calls to each other and share data structures, we believe they form a single program, which must be treated as an extension of both the main program and the plug-ins. This means the plug-ins must be released under the GPL or a GPL-compatible free software license, and that the terms of the GPL must be followed when those plug-ins are distributed.
>
> If the program dynamically links plug-ins, but the communication between them is limited to invoking the 'main' function of the plug-in with some options and waiting for it to return, that is a borderline case.[19]

[15] *See* GNU Operating System, All GNU Packages, (2013/06/17 04:56:31), http://www.gnu.org/software/software.html#allgnupkgs.
[16] *See* Frequently Asked Questions about version 2 of the GNU GPL, *What legal issues come up if I use GPL- incompatible libraries with GPL software* (2013/03/10 06:17:30), http://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.html#GPLIncompatibleLibs.
[17] *See If I modify GPL-covered programs …,* (2013/03/10 06:17:30), http://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.html#MoneyGuzzlerInc.
[18] *See, e.g.,* Lothar Determann, *Dangerous Liaisons—Software Combinations As Derivative Works? Distribution, Installation, And Execution Of Linked Programs Under Copyright Law, Commercial Licenses, And The GPL*, 21 BERKELEY TECH. L. J. 4, 1465 (2006).
[19] *See* Frequently Asked Questions about the GNU Licenses, *If a program released under the GPL uses plug-ins, what are the requirements for the licenses of a plug-in?* (2013/06/11 04:07:40), http://www.gnu.org/licenses/gpl-faq.html#GPLAndPlugins.

Software components written in Javascript and other scripting languages are referred to as "interpreted language programs," meaning that they are executed in source code form by a program called an "interpreter." These scripts are not compiled into object code. The interpreter interacts with an application program and thereby uses the scripts to extend or enhance the functionality of the application program. As a general matter, components written in Javascript and other scripting languages are neither statically nor dynamically linked to the application program. Therefore, most commentators agree that the use of these components does not result in derivative works in the context of the *GNU General Public Licenses*.

The FSF agrees with this general proposition, stating on its website that:

> "[t]he interpreted program, to the interpreter, is just data; a free software license like the GPL, based on copyright law, cannot limit what data you use the interpreter on. You can run it on any data (interpreted program), any way you like, and there are no requirements about licensing that data to anyone."[20]

However, the FSF is careful to point out that there are implementations where interpreted programs may effectively be linked, either statically or dynamically, to an interpreter or some of its components. For example, the interpreter may be extended to provide "bindings" to other components, such as libraries, which are used by the interpreted program. The FSF believes that these situations give rise to derivative works, because the interpreted program is effectively linked to components of the interpreter through the bindings. [21] Another example noted by the FSF is where libraries (*e.g.*, Java classes) provided with the interpreter are themselves interpreted. These libraries and the programs that call them (*e.g.*, Javascript programs) are always dynamically linked together. According to the FSF, these implementations result in derivative works that will be subject to the *GNU General Public Licenses* if the relevant components of the interpreter or libraries are themselves subject to those licenses.[22]

C. Linking Exceptions

The copyleft provisions of the *GNU General Public Licenses*, especially as interpreted by the FSF, can be extremely harsh when it comes to software components, like libraries, that are intended to be linked with other programs. Applying the *GNU General Public Licenses* to such components may therefore result in certain OSS components not being widely adopted and used because proprietary software producers might choose to avoid them in fear of their proprietary programs being subject to copyleft obligations as well. Therefore, providers of libraries and other linked components will often release their OSS under the *GNU Lesser General Public License*. This license agreement, which was designed by the FSF as an alternative to the *GNU General Public License*,[23] imposes copyleft obligations on the OSS components themselves, but not on the programs with which they are linked. The license agreement expressly provides that:

> [a] program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.[24]

Another compromise to the *GNU General Public License* has become known as the "classpath exception" or "linking exception," which is a paragraph added to the *GNU General Public License* to expressly exclude linked libraries from the scope of the license. The typical classpath exception paragraph is as follows:

> Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License cover the whole combination. As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you may extend this exception to your version of the library, but

[20] *See If a programming language interpreter is released under the GPL, does that mean programs written to be interpreted by it must be under GPL-compatible licenses?* (2013/06/11 04:07:40), http://www.gnu.org/licenses/gpl-faq.html#IfInterpreterIsGPL.
[21] *Id.*
[22] *Id.*
[23] *See Why you shouldn't use the Lesser GPL for your next library* (2014/04/12 12:39:52), http://www.gnu.org/licenses/why-not-lgpl.html.
[24] *See* Wilipedia, the Free Encyclopedia, *GNU Lesser General Public License* (17 Sept. 2014 at 19:50), http://en.wikipedia.org/wiki/GNU_Lesser_General_Public_License.

> you are not obliged to do so. If you do not wish to do so, delete this exception statement from your version.[25]

The classpath exception also allows a downstream user to link OSS libraries with proprietary software programs without the proprietary programs being subject to copyleft obligations. The classpath exception and the *GNU Lesser General Public License* lead to the same result; nevertheless, they are different in certain ways and should therefore be treated separately. For example, the *GNU Lesser General Public License* imposes restrictions on the OSS libraries that are not imposed by the *GNU General Public License with Classpath Exception*, such as the obligation to allow reverse-engineering and modifications and the obligation to provide source code along with any distribution of the OSS libraries.

IV. Making Source Code Available

As mentioned, some OSS licenses require that the source code of the OSS, and possibly works derived from it, must be made available to downstream users whenever there is a distribution of the licensed OSS in object code form (*i.e.*, compiled or executable form). The specific steps that must be taken to fulfill this obligation may differ based on the applicable OSS license agreement. The most common OSS license agreements, such as the *Common Development and Distribution Agreement, Version 1.0*; the *GNU General Public Licenses*; the *Mozilla Public License, Version 2.0*; or the *Eclipse Public License, Version 1.0*; generally provide the option to either include a copy of the source code to downstream users along with the distributed object code or to inform downstream users how they can obtain the source code.

One practical approach to publishing OSS source code is to post it on a public File Transfer Protocol ("FTP") server and to include general notices to customers or other recipients regarding how to access and download the source code from that server. This approach will satisfy the requirements of at least the most common OSS licenses, though the company will still need to determine whether the applicable OSS license agreements impose certain specific obligations, such as requiring source code to be available for a particular duration. For example, the *GNU General Public Licenses* provide licensees with the option to distribute source code by accompanying the object code with "a written offer, *valid for at least three years*, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed… on a medium customarily used for software interchange."[26] On the other hand, the *Common Development and Distribution License, Version 1.0,* states only that "[y]ou must inform recipients of any such Covered Software in Executable form as to how they can obtain such Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange."[27]

V. Attribution and Legal Notice Obligations

Most OSS license agreements include a clause requiring that all copies of the licensed OSS retain notices identifying the author(s) or original distributor(s) of the OSS, as well as any copyright, patent, trademark, or other legal notices found in the original version of the licensed OSS. Some OSS license agreements also require the licensee to provide notice of any modifications it made to the OSS when the licensee distributes such modifications. OSS license agreements may also require that distributions include a notice regarding and/or a copy of the OSS license agreement itself. These notice requirements may apply to distributions of source code and/or object code of the OSS, or any code derived from it.

In some cases, an OSS license agreement will specify the manner in which attribution and/or other notices must be provided (*e.g.*, in headers of source code files, in "help" or "readme" files, within graphical user interfaces). In cases where an OSS license agreement is silent as to the manner for providing required notices, many companies will do so by providing in their product documentation, or via an associated website, a listing of all third-party software included in their product, along with the applicable copyright and license information. In any event, these notice obligations are easily satisfied when an organization tightly manages and tracks its use of OSS such that compliance checks can be performed and/or approved by the legal department before a product ships. On the other hand, notice obligations can become extremely burdensome in cases where the legal

[25] *See* GNA Classpath, *Classpath License,* (2009/07/30 22:51:18), http://www.gnu.org/software/classpath/license.html.
[26] *See* Open Source Initiative, *GNU General Public License, version 2 (GPL-2.0),* (June 1991), http://opensource.org/licenses/gpl-2.0.php (emphasis added); *see also* GNU General Public License, version 3 (GPL-3.0) (29 June 2007), http://opensource.org/licenses/gpl-2.0.php (emphasis added); *see also* http://opensource.org/licenses/gpl-3.0.html. Note that the three-year obligation applies only to licensees who choose to comply with source-code distribution obligations by making such a written offer. The *GNU General Public Licenses* provide licensees with other options for complying with source-code distribution obligations.
[27] Common Development and Distribution Agreement, Version 1.0, section 3.1, http://opensource.org/licenses/CDDL-1.0, (last visited October 16, 2014).

department or compliance group is unaware of what OSS may be included in products that are about to ship or have already been shipped. For example, it is not uncommon for large-scale products and systems to include hundreds or even thousands of OSS programs, modules, libraries and the like.

VI. Implications of OSS Licensing for Cloud Computing

"Cloud" solution providers, also referred to as Application Service Providers ("ASP") or Software as a Service ("SaaS") solutions, do not primarily license software to customers. They operate on a subscription model, delivering services through an internet connection with most (though not necessarily all) functions performed on a server at a data center and with little, if any, software actually delivered to customers. This trend initially appeared as a potential threat to the growth of OSS; with the passage of time, OSS proponents have found ways to reach the expanding range of cloud services. For example, the *GNU Affero General Public License* provides that users who "interact" with software as a service in the cloud get the source code just as if they would if the software were subject to one of the *GNU General Public Licenses.*[28]

Furthermore, many high-value, high-functionality cloud solutions actually deliver software to the client computer as scripts that run in the browser, or in other packages that are installed at the client computer location. These deliveries are used to make the cloud services more efficient, such as plug-ins to common programs such as email software. Accordingly, such cloud services may ultimately be susceptible to the "copyleft" and other distribution requirements of OSS licenses, at least according to the FSF's view of derivative works.

VII. Patent Issues

Some OSS licenses include express or implied patent licenses or non-assertion covenants. A concern with these types of license provisions is that a company that distributes OSS under one of these licenses will be precluded from asserting its patents against downstream users. A similar concern arises with respect to a company's unrelated commercial patent licensing activities. For example, if a company licenses a competitor under certain of the company's patents and the competitor knowingly or unknowingly distributes OSS that includes functionality covered by the licensed patents, then there may be concern that the competitor has effectively granted royalty-free sublicenses under such patents to all downstream recipients. Likewise, if the competitor did not have the right to grant sublicenses under the patents, it may be in breach not only of the commercial patent license agreement but also of the OSS license agreement.

The *GNU General Public License, Version 3* and the *Apache License, Version 2.0* are examples of OSS license agreements that include express patent license provisions. Generally stated, under these agreements, each contributor of copyrightable content to the licensed OSS automatically grants to each downstream recipient a non-exclusive, worldwide, royalty-free license under all patents owned or controlled by the contributor that would otherwise be infringed by the recipient's making, using, or selling that version. The patent license granted by a contributor does *not* cover any further modifications that the recipient may make to the version acquired from that contributor. Accordingly, distributing software under the *GNU General Public License, Version 3* or the *Apache License, Version 2.0* does not have the effect of granting to the entire market a blanket license to all of a company's patents for any and all purposes. A patent license will apply only if the company contributes copyrightable content to the version of the OSS that it distributes — and even then it will apply only to the functionality included in that distributed version.

Like some other OSS license agreements with copyleft provisions, *GNU General Public License, Version 2* does not provide an express patent license, but arguably includes an implied patent license that is essentially the same as the one granted under the *GNU General Public License, Version 3* or the *Apache License, Version 2.0*. The implied patent license arguably results from the fact that the copyleft provisions require that any distribution of the OSS or derivatives of it must be under the same terms, which preserve the recipients' freedom to use, modify and redistribute the code. Therefore, it would be inequitable for a company to distribute OSS to which it has contributed functionality and then file a patent suit against any of the recipients for exploiting that functionality as authorized by the OSS license. Accordingly, to the extent these implied patent licenses exist, they should be fairly limited in scope.

[28] Version 3, (2013/02/28 17:09:28), http://www.gnu.org/licenses/agpl-3.0.html.

In the case where a company modifies some OSS to include functionality covered by patents licensed from a third party, the distribution of that modified OSS will typically result in a downstream patent sublicense only when the company has the right to grant sublicenses under the patents. For example, the patent license required under the *GNU General Public License, Version 3* extends only to those patents for which the company has the right to grant sublicenses in a manner consistent with that agreement. Similarly, the patent non-assertion covenant required under the *Apache License, Version 2.0* extends only to those patents licensable by the company. Nevertheless, the *GNU General Public Licenses* expressly prohibit the distribution of OSS where downstream users would not have the complete freedom to use, modify and redistribute it. The *GNU General Public License, Version 2* states that "… if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program."[29]

Therefore, to address concerns relating to OSS license agreements and unrelated commercial patent licensing activities, companies should ensure coordination between the relevant stakeholders. Companies should also consider including provisions in their commercial patent license agreements to prohibit sublicensing and/or to prohibit the licensee from including any of the patented functionality in any OSS components, at least without the licensor's consent.

VIII. Warranties and Limitations of Liability

Another significant risk associated with the use of OSS is potential liability for claims of patent or copyright infringement. OSS projects often have many contributors, any of whom could unknowingly contribute infringing code. Similarly, contributors to OSS projects make no commitments regarding the quality or fitness of their code. Furthermore, the typical OSS license agreement does not include any representations, warranties, or indemnities in favor of the licensee; to the contrary, the typical OSS license agreement will include a broad disclaimer of all representations and warranties that might otherwise be expressly or impliedly provided by a commercial software licensor.

IX. Corporate OSS Policies

Establishing a corporate policy regarding the management and use of OSS is generally considered best practice for companies that acquire or develop software. An open source policy should clearly specify the criteria to be used, as well as the people responsible for determining when and how OSS may be used within the organization. The policy should also establish processes and procedures for the intake, use and redistribution of OSS and for ensuring compliance with OSS licensing obligations. The policy document should be designed for distribution to software development teams and should be as concise as possible; an overly dense document will be unwieldy for busy software developers and is more likely to be marginalized.

An effective OSS management process will be simple and transparent. Software developers require quick responses to OSS-related questions and requests for approval regarding the use of OSS within or in conjunction with the company's software. A common approach is to identify in the policy document any particular OSS use cases that will not require additional approval from management or legal functions. For all other use cases, the policy should describe what information software developers must submit to the decision-makers to facilitate the approval process. To assist with the management of OSS, it may be desirable to designate an "owner" within the company (or within a particular business unit or product group) who will be knowledgeable about the company's use of particular OSS components and will be responsible for the company's compliance with the applicable OSS licenses. An OSS policy may provide for continuing education and training for company personnel in the handling of OSS and for periodic auditing of compliance with the OSS policy.

The OSS policy should specify the records that must be kept regarding each OSS component approved for use within the company, including the applicable OSS license, its source and maintenance history, where and how it is used within the company and any modifications made to it by the company. Disapproved OSS requests may also be tracked. These records can be used to manage the OSS life cycle and should be consulted by software developers and/or OSS managers during the approval process to ensure consistency with prior decisions. The repository should also store archived copies of all OSS acquired by the company — particularly those OSS components that have been modified by the company — so that new copies are not later acquired from other external sources. Ideally, the repository will correlate proprietary software for which the company has already

[29] http://opensource.org/licenses/gpl-2.0.php, (June 1991).

obtained a commercial license with equivalent OSS components, so that unnecessary use of OSS may be avoided.

X. OSS Audits

Many companies find themselves in the position of knowing that OSS is widely used by their software development teams, yet not having established a formal OSS policy or any OSS management procedures. A company in that position is unlikely to know exactly what OSS components are used within the organization or how they are used, let alone whether the company is in compliance with all applicable OSS license agreements. The only true way to establish a baseline — to identify all OSS used by the company and to address all known instances of noncompliance —is to conduct a full audit of the company's entire code base. OSS audits are typically initiated using automated code scanning and auditing tools. These tools may perform a variety of functions such as identifying OSS, applicable OSS licenses, license conflicts and copyright issues. Some of these tools are available from commercial providers, such as Black Duck[30] and Palamida,[31] and some are distributed as OSS programs.[32] Automated code-scanning and auditing tools generate reports that must be analyzed to filter out false-positives and detect true issues of noncompliance.

XI. Noncompliance and Remediation

Each piece of OSS that is found through an audit or is otherwise known to be used by a company may raise its own license-compliance issues. Depending on the size of the code base, the number of OSS and other third-party components can easily reach into the hundreds or thousands, which can require a massive coordinated effort to identify and remediate instances of noncompliance. To manage the remediation process, it is usually most effective to build a database of all relevant OSS components so they can be searched, sorted and prioritized according to risk level, license type and location at the company or within company products. Ideally, the database will track the names of relevant software developers, attorneys and others involved in evaluating each OSS component, as well as the status of such evaluation.

In order to reach a conclusion about compliance, it is usually necessary to consult with the software development lead, architect, system administrator or other power user who has actual knowledge of how the company has used each component and how that component may interact with the company's proprietary code. The most salient issues are: how the company uses an OSS component; whether the company has created any derivative works of the OSS; and whether the company has delivered that derivative program to a customer and, if so, under what license terms. With answers in hand, counsel can then look to the particular OSS license agreement to analyze whether such uses are permitted and whether the company has complied or will be able to comply with all relevant obligations imposed by the license agreement.

If the team conducting the compliance review process determines that OSS components are being used outside the scope of the applicable licenses, then some type of reasonable remediation effort should be undertaken (or at least considered) in view of the risks presented. In some cases, the required remediation effort will be simple and straightforward, such as releasing source code that the company failed to release when it distributed an OSS component. In other cases, the required effort may be impractical, extremely time-consuming and/or cost-prohibitive. In some cases, several different remedial options may be available.

In many situations, noncompliance with OSS license agreements primarily results from failing to include attribution and legal notices for OSS code snippets that are used within the company's code base. In some cases, a company can remedy its noncompliance simply by adding the required attribution text and legal notices to the appropriate source code files that it makes available (*e.g.*, via an FTP site) pursuant to the applicable release. Where the notices need to be included in files that have already been distributed to customers, the most practical approach may be to include the notices in subsequent releases of the software. Depending on the number of affected files, the company might consider sending notices to its customers and/or seeking waivers from the copyright holders regarding past noncompliance.

[30] Black Duck Software, http://www.blackducksoftware.com, (last visited October 16, 2014).
[31] Palamida, Inc., http://www.palamida.com, (last visited October 16, 2014).
[32] *See, e.g.,* Fossology, http://www.fossology.org/projects/fossology, (last visited June 25, 2013); Daniel M German and Yuki Manabe *Ninka, a license identification tool for Source Code,* (2011-01-22 00:40:15), http://ninka.turingmachine.org.

In cases where source code files are numerous, very large and have OSS code embedded throughout, it may not be a simple task to go back and add attribution and legal notices. One thing to keep in mind when faced with this situation is that some OSS license agreements are more specific than others when it comes to the manner in which attribution and legal notices are to be applied. For OSS license agreements that do not specify exactly how this should be done, a more practical approach may be to create a separate text file to be distributed along with the company's software product, which identifies:

- The name of each file that includes OSS code snippets and, if possible, the approximate location(s) within the file where the snippets can be found;
- A copy of each of the included OSS code snippets; and
- The applicable attribution and legal notices for the OSS code snippets.

Providing this information would allow downstream users to easily identify the OSS (which might otherwise be difficult to distinguish from the company's code) and could be considered a reasonable approach to compliance with attribution and legal notice obligations. This more global approach will still require significant effort, but may prove to be more efficient than a piecemeal, file-by-file approach.

Examples of more complicated instances of noncompliance include using OSS components governed by conflicting or incompatible OSS license agreements and/or using OSS components in such a way as to create derivative works subject to copyleft obligations (where the company has an aversion to releasing any of its proprietary source code). A license conflict arises when the conditions of one license make it impossible to comply with conditions of another license. A common license conflict occurs when an OSS license agreement mandates that downstream users have the freedom to reverse-engineer the OSS code, but the OSS code is bundled with proprietary software where the governing commercial license agreement prohibits reverse-engineering. Furthermore, note that many OSS license agreements are deemed to be incompatible with the copyleft provisions of the *GNU General Public Licenses*.[33]

Although removing and replacing troublesome OSS code is an option for remediating noncompliance for future software releases, a company will still need to consider noncompliance issues for previously released software. Again, the company will need to decide whether to bear the risk associated with past noncompliance, or instead spend considerable time and money to try to remediate the past noncompliance, such as by making patches available to customers, providing support or support technicians to customers to resolve the issue or, in the most extreme case, to recall the software. Where removing and replacing code from past or future software releases is prohibitively time-consuming and expensive, another option may be to seek a waiver or a commercial license from the author or licensor of the applicable OSS code. In addition, a check should always be made to determine whether the applicable OSS code has been released under another OSS license agreement (i.e., dual-licensing) that either does not present any conflicts or does not include unwanted copyleft provisions.

XII. OSS in Corporate Transactions

The need to manage OSS risk in corporate transactions such as mergers and acquisitions, commercial licensing deals and the like, should not be underestimated. Partnering with another company that has failed to properly manage its use of OSS can subject a company to unanticipated risks and liabilities. It is important for any company preparing for such a transaction to spend the time and money necessary for an adequate pre-transaction due diligence study. The scope of such due diligence will naturally depend on the size and nature of the contemplated transaction. On the one hand, a corporate acquisition with a price tag of several hundred million dollars will mandate a significant and thorough due diligence effort by the purchaser, likely involving a full OSS audit of the company to be acquired. On the other hand, a deal valued at few hundred thousand dollars may require a less intensive and more focused due diligence investigation.

In almost all corporate transactions involving software assets, the contractual representations, warranties and indemnities relating to OSS will be heavily negotiated. Both parties will attempt to shift as much risk regarding OSS noncompliance or unwanted copyleft obligations to the other party. For instance, in most mergers and acquisition agreements, the acquiring party will likely push for a sweeping definition of "Open Source Software," and seek a representation and warranty that the acquired software or business includes absolutely no "Open Source Software" except as disclosed in a schedule of exceptions. If a full OSS audit is not practical or reveals

[33] *See* Various Licenses and Comments About Them, *GPL-Incompatible Free Software Licenses,* (2013/04/11 13:34:23), http://www.gnu.org/licenses/license-list.html#GPLIncompatibleLicenses.

issues that cannot be remediated prior to closing the contemplated transaction, then the party acquiring or licensing the software assets or business should consider seeking additional contractual protections such as indemnities, a cash escrow, or hold-back to cover the cost of any remediation efforts that may be necessary after closing. From the seller’s or licensor’s perspective, it may not be practical to accept much (or any) of the risk associated with OSS.

Sellers and licensors seldom, if ever, have a complete list of OSS. Even well-managed companies tend to have imperfect or somewhat outdated OSS tracking. But even an incomplete list can be burdensome to prepare and take up dozens of pages in the disclosure schedules. For buyers, receiving the OSS disclosure list impels the team to determine if any of the disclosed OSS is truly a “deal killer.” Law firms and consulting firms will often be called upon to handle this situation on both sides of the transactions. Many of the risk criteria discussed in this article can serve as a guide to prioritizing issues and communicating risk levels in the context of due diligence and analysis of disclosures against representations.