Free and Open Source Software Compliance

The Basics You Must Know

By Ibrahim Haddad, Ph.D.

JUNE 2010

Free and Open Source Software Compliance *The Basics You Must Know*

By Ibrahim Haddad Ph.D.

THE LINUX FOUNDATION

Executive Summary

This white paper is a first in a series that will focus on the various practical aspects of ensuring free and open source software (FOSS) compliance in the enterprise. This paper provides basic discussion on the following topics:

- The changing business environment moving to a multi-source development model
- The objectives of compliance and the benefits resulting from having a successful compliance program
- The consequences of non-compliance with the licenses of free and open source software
- The compliance failures that can occur, how to avoid them and prevent them from happening in the future
- The lessons learned from the various non-compliance cases with emphasis on the positive learnings

A Changing Business Environment

Traditionally, platforms and software stacks were implemented using proprietary software and consisted of various software building blocks that came from different 3rd party software providers with negotiated licensing terms (Figure 1). The business environment was predictable and companies mitigated potential risks through license and contract negotiations with the software vendors.

Applications (3rd Party Commercial / Proprietary)
Middleware (3rd Party Commercial / Proprietary)
Operating System (3rd Party Commercial / Proprietary)
Drivers (3rd Party Commercial / Proprietary)
H/W Chips (Multiple Vendors)

- Commercial licenses are negotiated
- There is a limited number of licenses
- The business environment is very predictable
- Companies ensure contractual protection through their commercial contracts and licenses
- There is a clear separation between the different components and their providers
- Risks are mitigated through license negotiation
- The providers of each software component are known

FIGURE 1. ILLUSTRATION OF THE ARCHITECTURE OF A TRADITIONAL SOFTWARE PLATFORM THAT RELIES ON PROPRIETARY AND 3RD PARTY COMMERCIALLY- LICENSED SOFTWARE BUILDING BLOCKS

With time, companies started to incorporate FOSS into their platforms for the different advantages it offers such as technical merit, fast time-to-market and the ability to customize source code to own needs. The current most adopted development model is the multi-source development model (Figure 2). In this model, software components can consist of source code origination from different sources and licensed under different licenses; for instance, software component A can include proprietary source code in addition to 3rd party proprietary source code, while software component B can include proprietary source code in addition to source code from an open source project. Figure 2 highlights the combinations of incoming source code sources.

Following this model, the source code built into a product is incoming from various sources:

- Proprietary, developed by the company building the product or by one of its acquired subsidiaries
- 3rd party commercial, developed by 3rd party software providers and received by the company building the product under a commercial license
- FOSS, developed by the FOSS community and received by the company building the product under a FOSS license
- A combination of any of the above sources

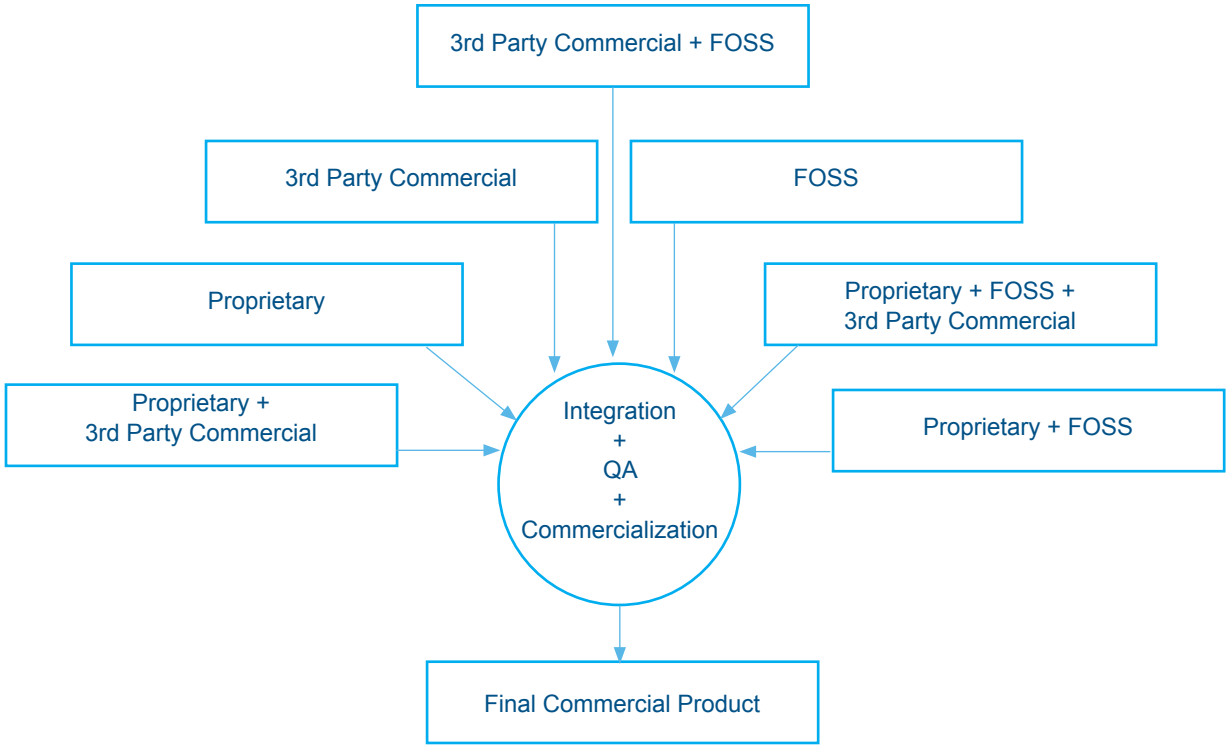
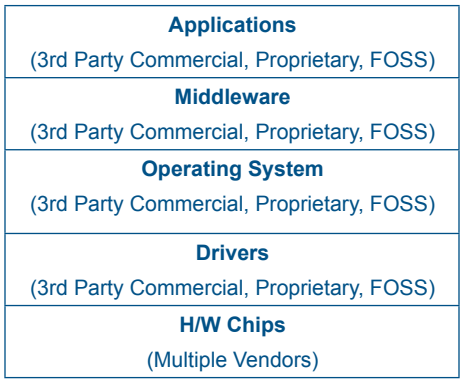


FIGURE 2. MULTI-SOURCE DEVELOPMENT MODEL

With the introduction of FOSS to what once were pure proprietary software stacks, the business environment diverged from familiar territory and corporate comfort zones (Figure 3). The licenses of FOSS are not negotiated agreements. There are no contracts signed with the software providers (i.e., FOSS developers). Companies must now deal with dozens of different licenses, and hundreds or even thousands of licensors and contributors. As a result, the risks that companies used to manage through license negotiations are now managed through compliance and engineering practices.



- Licenses are not negotiated; they are imposed
- There are potentially tens of licenses involved
- The business environment is not as predictable as in a purely commercial environment
- There are potentially thousands of contributors to the various FOSS projects used
- The origin of some of the components may not be clear
- Risks are mitigated through design, engineering practices and compliance

FIGURE 3. ILLUSTRATION OF THE ARCHITECTURE OF A MODERN SOFTWARE PLATFORM SHOWING THE PROLIFERATION OF FOSS INSIDE EACH OF THE PLATFORM SOFTWARE BUILDING BLOCKS

> *In fact, in nearly every GPL enforcement case that I've worked on in my career, the fact that infringement had occurred was never in dispute. The typical GPL violator started with a work under GPL, made some modifications to a small portion of the codebase, and then distributed the whole work in binary form only. It is virtually impossible to act in that way and still not infringe the original copyright.*
>
> *– Bradley M. Kuhn, Policy Analyst and IT Director, Software Freedom Law Center (11/09/2009, http://www.softwarefreedom.org/blog/?tag=infringement)*

Enter FOSS Compliance

FOSS initiatives and projects provide companies with a vehicle to accelerate innovation through collaboration with the global community of FOSS developers. However, accompanying the benefits of teaming with the FOSS community are important responsibilities: Companies must ensure compliance with applicable FOSS license obligations.

FOSS compliance means that users of FOSS must observe all the copyright notices and satisfy all the license obligations for the FOSS they use. In addition, companies using FOSS in commercial products, while complying with the terms of FOSS licenses, want to protect their intellectual property and that of 3rd party suppliers from unintended disclosure.

Compliance helps achieve four main objectives:

1. **Comply with FOSS licensing obligations**

2. **Facilitate effective usage of FOSS in commercial products**
3. **Comply with 3rd party software supplier contractual obligations**
4. **Protect commercial product differentiation**

Benefits of Ensuring FOSS Compliance

There exist several side benefits to achieving compliance that includes:

- Gaining a technical advantage since compliant stacks are easier to service, upgrade, and test
- Gaining an increased understanding of the benefits of FOSS and how it impacts your organization
- Gaining an increased understanding of the costs and risks associated with using FOSS
- Gaining an increased knowledge of available FOSS solutions
- Building a relation with the FOSS community and FOSS organization through involvement with FOSS projects and participation in FOSS events
- Better corporate readiness in preparation for possible acquisition, sale, new product or service release, where compliance assurance is mandatory before the completion of any of these transactions. Furthermore, there is the added advantage of verifiable ensured compliance in dealing with OEMs and downstream vendors
- Improving your overall FOSS strategy using the results from your compliance program

Failure to Comply

Several failure and consequently to the failure of complying with the FOSS license obligations, for instance, in regard to:

- Recognition of:

    1. Attribution notice: An attribution notice is a notice included in the product documentation that acknowledges the identity of the original authors of the FOSS included in the product.

        Example of an attribution notice (for Webkit1):

        Contributors to the WebKit, WebCore and JavaScriptCore projects include: Alex Mathews, Alexander Kellett, Alexey Proskuryakov, Allan Sandfeld Jensen, Alp Toker, Anders Carlsson, Andrew Wellington, Antti Koivisto, Apple Inc., Bjoern Graf, Brent Fulgham, Cameron Zwarich, Charles Samuels, Charlie Bozeman, Christian Dywan, Collabora Ltd., Cyrus Patel, Daniel Molkentin, Daniel Veillard, Dave MacLachlan, David Smith, Dawit Alemayehu, Dirk Mueller, Dirk Schulze, Don Gibson, Enrico Ros, Eric Seidel, Frederik Holljen, Frerich Raabe, Friedemann Kleint, George Staikos, Google Inc., Graham Dennis, Harri Porten, Henry Mason, Hiroyuki Ikezoe, Holger Hans Peter Freyther, International Business Machines Corporation, James G. Speth, Jan Michael C. Alonzo, Jean-loup Gailly, Jon Shier, Jonas Witt, Julien Chaffraix, Justin Haygood, Kevin Ollivier, Kevin Watters, Kimmo Kinnunen, Kouhei Sutou, Krzysztof Kowalczyk, Lars Knoll, Luca Bruno, Lucent Technologies, Maksim Orlovich, Malte Starostik, Mark Adler, Martin Jones, Matt Lilek, Michael Emmel, Netscape Communications Corporation, Nicholas Shanks, Nikolas Zimmermann, Nokia Corporation, Nuanti Ltd., Oliver Hunt, OpenedHand, Peter Kelly, Pioneer Research Center USA, Inc., Rob Buis, Robin Dunn, Ronald Tschalär, Samuel Weinig, Simon Hausmann, Staikos Computing Services Inc., Stefan Schimanski, Symantec Corporation, The Karbon Developers, Thomas Broyer, Tim Copperfield, Tobias Anton, Tony Chang, Torben Weis,

Trolltech ASA, University of Cambridge, Vaclav Slavik, Waldo Bastian, Xan Lopez, Zack Rusin, mozilla.org.

2. License notice: A license notice is a notice that acknowledges the license terms and conditions of the FOSS included in the product.

   Example of a license notice (for rsync2):

   This software package is licensed under the GPL version 2. See COPYING file for the full GPL license text.

   or

   This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA

3. Copyright notice: A copyright notice is an identifier placed on copies of the work to inform the world of copyright ownership.v

   Example of a copyright notice (for the libpcap3 library):

   Copyright © The Internet Society (2004). All Rights Reserved.

4. Modification notice: A modification notice is a notice of the modifications made to the source code in a change log file, such as those required by the GPL and LGP.

   Example of a modification notice:

   /*

   * Date Author Comment

   * 01/05/2010 Ibrahim Haddad Fixed memory leak in getnext()

   */

- Making inappropriate or misleading statements in the product documentation or product advertisement material
- Making available:

1. The software source code that corresponds to the binaries distributed in the shipping product. In some instances, companies may not inform users on how they can obtain the source code for software packages included in the product. This is often referred to as written offer.

   Example of a written offer to provide the source code:

   To obtain a copy of the source code being made publicly available by FooBar, Inc. related to software used in this FooBar product, you can visit http://opensource.foobar.com or send your request in writing by email to opensource@foobar.com or by snail mail to:

   FooBar Inc., Open Source Program Office
   Street Address
   City, State, Postal Code
   Country

[1] Available from http://webkit.org

[2] Available from http://samba.anu.edu.au/rsync

[3] Available from http://www.tcpdump.org

The build scripts needed to compile the source code.

Table 1 illustrates the most common compliance failures that occur during the software development process, a brief description of the failure and how it occurs, in addition to measures that a company can implement to avoid and prevent such failures from happening again.

TABLE 1. EXAMPLES OF COMPLIANCE FAILURES AND HOW TO AVOID THEM

FAILURE TYPE AND DESCRIPTION	POLICIES AND PROCEDURES TO AVOID SUCH FAILURES
Unplanned or unapproved inclusion of FOSS into proprietary or 3rd party code or vice versa (i.e. failure to submit an Open Source Review Board (OSRB) request form): These failures occur during the software development process when engineers add FOSS code into proprietary or 3rd party source code without having the necessary required approval from the OSRB. These failures can be discovered by: • Scanning[4] the source code for possible matches with FOSS source code. • Implementing and running a bill of materials[5] (BoM) difference tool[6] to identify source code modifications at product level and component level.	• Offer training to bring awareness to compliance issues and to the different types and categories of FOSS licenses and the implications of including FOSS source code in proprietary or 3rd party source code without proper approval. • Conduct regular source code scanning all the source code in the code base to discover source code additions that do correspond to a compliance ticket[7]. • Mandate engineers to fill out OSRB forms[8] to inform the OSRB of their intention to use FOSS source code in a proprietary or 3rd party software, or as a standalone. • Include compliance goals in the employees performance review. Following this method, failure to abide by the compliance policies will affect directly the employee's bonus reward; as a result, engineers will have additional incentives to ensure compliance. • Mandate that for a FOSS to be accepted into the build system, it must have a corresponding OSRB form.

[4] The are several commercial and FOSS tools that offer the capabilities of scanning source code and identifying what FOSS has been discovered in the source code base. We will be discussing these tools in a future paper.

[5] The bill of material (BoM) is the list of software components included in the final shipped product.

[6] The BoM difference tool takes as input two BoMs, computes the differences, and outputs the changes found between the two BoMs. The output highlights new software components that entered the source code base, retired software components that are no longer used, and software components that were modified.

[7] A ticket in this context is similar to task in a project management tool. Each FOSS must have a corresponding compliance ticket. The goal is to ensure that for any specific FOSS, the company knows how this FOSS is being used, in which product(s) it is included, and has a plan in place to meet the license obligations.

[8] The form is a questionnaire that engineers fill out and submit to the Open Source Review Board (OSRB) requesting approval to use the FOSS in question in a particular product. The OSRB reviews requests for use, modification, and distribution of FOSS and determines approval. In addition, the OSRB serves as steering committee to define and manage the company's FOSS strategy

Linking[9] of FOSS into proprietary or 3rd party commercial source code (or vice versa): This failure occurs when linking software (FOSS, proprietary, 3rd party) that have conflicting and incompatible[10] licenses, or if the FOSS has a license with a viral effect[11]. This type of failure can be is discovered using a dependency tracking tool[12] that allows you to discover linkages between different software components.	Offer training to avoid linking software components with conflicting licenses or to those software components whose license has a viral effect • Continuously run the dependency tracking tool over the build environment to verify linkages and to flag any issues. • Mandate engineers to disclose linkage method and list of linked components as part of the OSRB form. This will provide the OSRB an early warning of any possible linkage problem.
Failure to publish source code: This failure occurs when using FOSS in a commercial product and not publishing the FOSS source code and modifications made to it as part of meeting license obligations (for instance when the FOSS is licensed under the GPL).	• Add a checklist item for source code publishing in the product release.
Failure to provide the right source code version: This failure occurs when making available the wrong version of the source code used to compile into a FOSS binary work that has been distributed as part of a product.	• Add a verification step into the compliance process to make sure you are publishing the exact version of source code that corresponds to the distributed binary in the product.
Failure to provide the modifications applied to the original FOSS: This failure occurs when the company fails to publish and/or mark the modification introduced to the FOSS that was distributed as part of a product.	• Add a milestone in the compliance process to verify that modified source code has been marked as such. • Conduct source code inspections before releasing the source code. • Offer training to engineers to ensure they modify the change logs of software that will be released to the public.

[9] Linkage in this context refers to dynamic and static method s of linkage.

[10] License compatibility is a term that refers to the problem encountered when combining source code originating from different software components licensed under incompatible licenses making it impossible to combine the source code to create a new software component. For more information on the concept of license compatibility, please refer to http://www.fsf.org/licensing/licenses/

[11] The term “viral effect” is a commonly and informally used term to describe the effect that copyleft licenses have where any works derived from a copyleft work must themselves be copyleft when distributed.

[12] The dependency mapping tool (also called dependency analysis tool or dependency tracking tool) takes as input the name of a software component and outputs the list of libraries linked to that software component via the dynamic and static linkage methods.

Failure to take the FOSS training: This failure occurs when employees neglect to take the mandatory FOSS training. As a result, they continue to be unaware of the company's policies and procedures with respect to FOSS and compliance.	• Mandate engineers to take the FOSS training by a specific date. • Ensure that the FOSS training is part of the employee's professional development plan.
Failure to audit the source code: This failure occurs when the OSRB fail to audit the source code incoming into the build system.	• Conduct periodic source code scans/audits. • Ensure that source code auditing is a milestone in the iterative development process. • Provide adequate resources to the OSRB to ensure that the compliance activities do not fall behind the development activities
Failure to resolve the audit findings: This failure occurs when the OSRB fail to finalize the audit results of any specific component, produce an audit report and attach it to the compliance ticket for that specific component.	• Implement a policy in the compliance management system that does not allow a compliance ticket to be closed if it has open sub-tasks or open issues.

Lessons Learned From Compliance Disputes

In the past few years, several cases of non-compliance found their way to the public. By examining these cases, we can extract the following lessons:

Lesson #1: Ensure Compliance Prior to Product Shipment

One common result from all the incompliance cases was that the company with the violation had to comply with the license. Therefore, it is recommended to ensure compliance prior to product ship.

It is important to acknowledge that compliance is not just a legal exercise. All facets of the companies are typically involved in ensuring proper compliance and contributing to the end-to- end management of FOSS. This includes establishing and maintaining consistent compliance policies and procedures, and ensuring that the licenses of all the software components in use (proprietary, 3rd party and FOSS) can co-existence well before shipment. To that effect, companies need to implement an end-to-end FOSS management infrastructure that will allow them to:

- Identify all FOSS used in products
- Perform architectural reviews to verify if and how FOSS license obligations are extending to proprietary and 3rd party software components
- Collect the applicable FOSS licenses for review by the legal department

- Develop FOSS use and distribution policies and procedures
- Mitigate risks through architecture design and engineering practices

We will be discussing the topic of implementing a company-wide compliance program in a future paper.

Lesson #2: Non-Compliance is Expensive

All of the disputes reached a settlement agreement that included one or more of the below mentioned terms:

- Company to take necessary action to become compliant
- Company to appoint a compliance officer to monitor and ensure GPL compliance
- Company to notify previous recipients of the product that the product contains GPL code and inform them of their rights to receive a copy of the source code
- Company to publish licensing notice on their website
- Company to provide additional notices in product publications
- Company to make available the complete and corresponding source code used in their product freely available on its website
- Company to cease binary distribution of the FOSS in question until it has published complete corresponding source code on its web site
- Company to pay an undisclosed amount of financial consideration to the plaintiffs

The FSF Compliance Lab: The Free Software Foundation Compliance Lab handles all licensing-related issues for FSF. They serve the free software community by providing the public with a “knowledge infrastructure” surrounding the GNU GPL and free software licensing, and enforcing the license on FSF-copyrighted software. (http://www.fsf.org/licensing)

Furthermore, the companies whose compliance has been successfully challenged have incurred costs that included:

- Discovery and diligence costs in response to the compliance inquiry, where the company had to investigate the alleged inquiry and perform due diligence on the source code in question
- Settlement costs with the plaintiffs
- Outside and in-house legal costs
- Possible damage to brand, reputation, and credibility

In almost all cases, the failure to comply with the FOSS license obligations has also resulted in public embarrassment, negative press and damaged relationships with some of their customers, suppliers and most notably the FOSS community (their software provider) and FOSS organizations.

> *Our number one goal in any GPL violation case is to get proper and full compliance with the license; everything else is secondary.*
>
> *– David Turner, GPL Compliance Engineer, Free Software Foundation (09/23/2003, http://lwn.net/Articles/51570)*

Lesson #3: Relationships Matter

For companies using FOSS in their commercial products, it is recommended to create and maintain a good relationship with the FOSS community. The community provides such companies with source code, technical support, testing, and documentation. Therefore, it is an expectation that these companies honor the licenses of the FOSS components included in their products. Taking steps in this direction, combined with open and honest relationship, is a great asset to have and maintain especially in the event that a company get into a compliance "oops".

Practical Guide to GPL Compliance: On August 26, 2008, the Software Freedom Law Center published a guide on how to be compliant with the GNU General Public License (GPL) and related licenses. The guide focuses on avoiding compliance actions and minimizing the negative impact when enforcement actions occur. (http://softwarefreedom.org)

Lesson #4: Training is Important

Training is an essential building block in a compliance program to ensure that employees have a good understanding of the policies governing the use of FOSS. All personnel involved in the development, QA, release and maintenance of software, need to understand the compliance program and their company's policies and procedures. Companies often provide such education through formal and informal training sessions.

Conclusion

This paper provided an overview of the changing business environment requiring a compliance program to ensure that companies honor the license obligations resulting from the use of proprietary, 3rd party commercial and FOSS components in their products. Furthermore, the paper highlighted examples of compliance failures, provided recommendations on how to avoid them and prevent them from happening again. The paper concluded with a number of lessons learned from the various non-compliance cases.

Future planned papers will discuss topics such as:

- FOSS licenses, obligations and how to ensure obligations are met
- Challenges facing enterprises in establishing and achieving compliance and how to overcome them
- The role and responsibilities of the teams involved in achieving FOSS compliance
- An overview of a sample end-to-end compliance management process
- FOSS compliance best practices
- Working with compliance inquiries

FOSS compliance is an essential part of the software development process. If you use FOSS in your product(s) and you do not have a solid FOSS compliance program, then you should consider this paper as a call to action.

Resources

The Linux Foundation	http://www.linuxfoundation.org
The Software Freedom Law Center	http://www.softwarefreedom.org
The Free Software Foundation	http://www.fsf.org
The GNU Project	http://www.gnu.org/licenses/gpl-violation.html

Acknowledgements

The author would like to express his gratitude to Karen Copenhaver (Legal Director of the Linux Foundation and Partner in Choate, Hall & Stewart LLP ‘s Business & Technology practice) and Eben Moglen (Professor of Law and Legal History at Columbia University Law School and Chairman of the Software Freedom Law Center) for their reviews and valuable input.

About the Author

Ibrahim Haddad is Director of Technology and Alliances at the Linux Foundation focusing on Mobile Linux initiatives and advancing the Linux platform for next-generation mobile computing devices.