Open Source License Enforcement and Compliance

Richard Fontana
Open Source Licensing and Patent Counsel
Red Hat, Inc.

Practising Law Institute
December 2009

Understand the Culture

- Source code sharing as old as the software industry
- Free software licensing emerges in early 1980s
- Evolving community norms/expectations of software-sharing conduct
- Effective open source compliance requires:
  - Understanding the **culture and traditions**
  - Understanding the **technology**

Understand the Boundaries

- Legal: perpetual royalty-free copyright license (termination for violation of conditions)
  - Essentially unlimited rights to (non-publicly) run, copy, modify
  - Distribution (and sometimes other public use) limited only in ways not customarily regarded as unduly burdensome to software freedom (e.g. notice, copyleft requirements)
- Technological
  - Source code, or object code with corresponding sources readily available to distributees

Understand the Boundaries

- Determined through community debate and community development and distribution practices
- Influential sources of authority: FSF, Debian, OSI, Fedora
- Hundreds of distinct FOSS licenses, but recent standardization trend
- You should adopt strict community-based standards for what is/isn’t authentic open source

Know the Basic License Categories

- Copyleft (strong, weak): limits freedom to use more restrictive outbound licenses
  - Usually some source code disclosure requirement
- Permissive/Non-Copyleft

Strong Copyleft (GPL)

- Policy goal: preserve a commons of free software (even as software evolves through downstream modification)
- “Strong” - licensor expectation that copyleft encompass all enhancements to the original work regardless of packaging
- Requirements:
  - Distribution of modified version must be under GPL
    - Exception for “mere aggregation”

Strong Copyleft (GPL)

- No imposition of “further restrictions” on downstream users’ exercise of GPL rights
  - Corollary: liberty-or-death clause
- Binaries: source code disclosure
  - “Complete”, “corresponding”, licensed under the GPL
  - Amount of source code ≈ scope of copyleft
    - System library exception
  - What a skilled programmer needs to rebuild the binary

Strong Copyleft Scope

- Clearest-cut case is statically-linked executable
- More difficult scope questions arose as technological circumstances changed (= new distribution fact patterns), motivation to circumvent GPL grew, and FOSS licensor/user community diversified
  - Heuristics developed through community debate
  - FSF/Stallman/Moglen as persuasive authority on GPL interpretation
  - Dissenters opt for non-GPL licensing models
- Largely academic issue for most distributors

Weak Copyleft

- Originates in community criticism of strong copyleft
- LGPLv2.x, MPL & progeny, CPL/EPL, LGPLv3
- Common features:
  - Copyleft scope (including source code requirement) limited to some subset of what is assumed to apply in GPL case (in practice, the source file)
  - Executables can be distributed under proprietary terms
- Wide gap between LGPL text and more liberal customary interpretation (e.g.: reverse engineering clause; GPL upgrade clause)

Permissive/Non-Copyleft

- BSD family, MIT family, Apache License 2.0, etc.
- Reaction against strong copyleft; continues older public domain tradition
- Policy goal: Maximize downstream adoption
- Derivative works licensable under more restrictive terms
- No source code requirement, but often a strong social expectation to contribute some improvements upstream

Compliance

- Make an effort to understand the **reasonable customary expectations** of particular licensors of particular works of software
- Avoid forcing community-developed licensing traditions into ill-fitting legal frameworks based on proprietary licensing models

Two Types of Vendors

- FOSS community **outsiders**
  - Do not contribute upstream
  - May not know whether products contain GPL'd code
  - Likely enforcement target if material GPL violations
- FOSS community **insiders**
  - Good faith effort to comply with license requirements
  - Develop good relationships with upstream communities
  - Unlikely to be enforcement target

Enforcement Risk

- Usually minimal to zero
- Heightened risk involves simple, avoidable fact pattern
  - Embedded device vendor sources firmware from third-party, which incorporates GPL'd code with at most minor modifications; no disclosure or due diligence
  - Invariably includes code copyrighted by small set of active GPL "community" enforcers (FSF, Welte, Andersen/Landley [BusyBox])
  - Material violation is failure to provide source code
  - The firmware is generally not a valuable IP asset

Enforcement Risk

- Other area of heightened risk: commercial licensor with dual-licensing business model
  - May be inclined to apply noncustomarily restrictive reading of GPL
- Vast majority of open source licensors expect voluntary compliance by downstream, but do not actively enforce

Product Development

- Compliance is usually easy once you know the applicable license terms
- Vendors must adopt careful third-party procurement practices (not just an open source issue), applied early in the development process
  - Don’t remediate close to product launch
- Good software engineering practices lead to good compliance
  - Know exactly what sources were used to build a particular binary – version control is essential

Due Diligence for Inbound Code

- Necessary regardless of how procured, and whether or not you think it is open source, and regardless of whether intended for product development
- Require upstream suppliers to provide:
  - Complete list of governing licenses
  - Complete list of *their* upstream suppliers
  - Explanation of procedures for open source compliance
- Consider shifting legal risk to the supplier

Due Diligence for Transparently FOSS Code

- Require developers to report what upstream source code they are including in the product
- Require developers to build from source
  - Significant cultural challenges in Java development
- You should be able to reconstruct how the code was put together and where it came from

Source Code Analysis

- Understand how legal information is customarily recorded and presented by upstream projects
- Identify external dependencies (e.g. shared libraries)
- Understand software build practices
- Know how to unpack a source tarball and browse a source tree (use ‘grep’)
- Use VCS to determine who committed what (do not rely on source files for authorship information)

Compliance Mechanics

- Once you know applicable licensing terms, take steps to ensure adequate compliance – usually easy if good practices have been followed
  - Notice requirements: for binary distributions, maintain a text file that contains all required legal notices
  - Source code requirements (for copyleft licenses)

Source Code Requirements: GPL

- Generally, 3-year written offer or accompany source with binary
  - Use latter, unless source distribution cost is true per-unit cost – e.g. embedded product with limited storage)
  - Don’t use written offer to postpone dealing with the problem!
  - FSF: for network distribution, can point to a location maintained by a third party (GPLv3 makes explicit)
- Must include build scripts and build instructions
  - GPL does not require “self-hosting”, but should provide information on what compiler was used

Source Code Requirements: Other Licenses

- LGPL: Can generally follow GPL rules; “suitable shared library mechanism” eliminates source requirement
- Other weak copyleft licenses: less detailed; assume written offer option not available
  - MPL-like licenses specify minimum post-binary-distribution time intervals

GPLv3 “Installation Information” Requirement

- Applies to binaries distributed in locked-down consumer products if the GPLv3’d software is modifiable by a third party
- Vendor must provide information sufficient to allow skilled developer to install functioning modified versions on the same device, with some limits
- No known enforcement experience
- Remember, restoration of rights under **GPLv2** may be conditioned on providing such unlocking information

GPL and Termination

- GPLv2 features automatic termination; key tool for active GPL enforcers in US and Germany
- GPLv3 provides two explicit cure opportunities: permanent restoration of rights if:
  - no complaint 60 days after coming into compliance
  - cure within 30 days of receipt of notice of first-time violation
- Enforcement targets may wish to take “GPLv2 or later” code as GPLv3 to invoke cure permission

Thank you

rfontana@redhat.com