Planning vs. Scanning

Considerations for the effective management of third-party software components

Although code scanning is an optimal technique for detecting the presence of open source content in sight-unseen code, it is not an effective overall approach to manage the ongoing incorporation of open source and commercial content in your software products.

The key to low cost, low risk leveraging of third-party software components ("Components") is not post-inclusion discovery, but rather moving the decision-making process to the earliest possible point in the product development lifecycle. Early review and assessment balances the engineering benefit of including such Components in your software products with the substantial and disruptive cost of removing code in the event that a post-inclusion discovery determines that the inclusion was noncompliant.

Prior to including any Component (open source *or* commercial) with your software product, it is essential that there is a review of the associated license *in the context of the engineering use case* by your legal department, finance department, and architectural stakeholders. Scanning simply cannot accomplish this.

Post-inclusion discovery has the following potential consequences, each of which the author has seen affect commercial software companies, and each of which can be avoided with proper planning and policy:

- Unscheduled stop ship of a high revenue product
- Engineering all-stop to remove inappropriately licensed content
- Declaration of material weakness by internal auditors resulting in loss of company stock value
- 6 and 7-figure legal settlements related to noncompliant shipment of commercially licensed content
- Disruptive audits by third-party commercial vendors
- Inadvertent open sourcing of valuable trade secrets
- Substantial external legal fees incurred to develop ad-hoc legal opinions that attempt to justify the Component use as compliant

Post-inclusion code scans can discover the presence of *open source* Components. However, these scans do not have a straightforward method to discover *commercial* Components. In addition, the scans do not have a method to determine the all-important *use case* of each Component.

OPEN SOURCE COMPONENTS

Most licenses associated with open source Components must be reviewed in the context of use – it is typically *not* the case that a company can say that *all* uses of, for instance, LGPL, Mozilla or GPL-licensed Components are, or are not, acceptable. Recording the use case at the time the Component is reviewed for inclusion in your product will allow you to track valuable information that is not discoverable in subsequent code scans.

The following are some examples where the use case of open source Components is critical:

- **LGPL**: LGPL-licensed Components are typically acceptable for inclusion in your commercial products as long as the following conditions are met: Components are accessed via a link, new versions of the library can be linked with the application, your license allows for modifications to the library itself and reverse engineering of your interface to the library to the extent required to replace a library, and you provide the source code for the library with the distribution of your commercial product. The same LPGL Components, however, may *not* be commercially acceptable if the LGPL code is included in-line or other requirements of the license are not met. Certain non-compliant uses of LGPL Components may cause the infringing product to be effectively governed by the GPL license, and a copy of the associated commercial product's source code may then be requested by virtually anyone.

- **Mozilla**: Software products that include Mozilla-licensed Components need to be offered under the Mozilla license unless the individual source code files have been kept separate from your product's source code files. The only way to determine if the files have been kept separate is to review the source code file structure, prior to Inclusion, with the engineer.

- **GPL**: Depending on the policy at your company, GPL-licensed Components may be included in your products *only* if they run as a separate process, are accessed only via command lines, pipes or sockets, *and* do not interact with your proprietary code in any detailed fashion. Also, some software companies allow GPL-licensed Components to be used in *hosted* environments as long as the hosted environment is not shipped to a third party, such as an ISV partner for outside hosting.

- **Eclipse vs. CPL**: The Eclipse 1.0 license is predated by the CPL (Common Public License). The CPL license terminates patent licenses granted via the CPL if a recipient institutes patent litigation against a contributor. The Eclipse 1.0 license, however, limits this patent termination to recipients that allege that the Component *itself* infringes on the recipient's patent. Such license distinctions are essential when selecting Components with licenses that match the risk profile of your company.

- **BSD (The "Original" 4-Clause License)**: The Berkeley Software Distribution (BSD) family of licenses is typically considered commercial-friendly. However, the original BSD license includes the following term: "All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the <organization>." This clause requires commercial software vendors to include acknowledgement of the original source in all advertizing materials – this is anathema to most marketing organizations, but must be done if the Component cannot be removed prior to product release[1].

Is your enterprise in compliance?

[1] Long time followers of the BSD-4-Clause license will note that the third clause (the "advertizing clause") was retroactively rescinded (see bottom of document) for Berkeley Software Distribution source code files by William Hoskins, University of California, Berkeley, Director, Office of Technology Licensing. However, the BSD-4-Clauuse license has been used by many copyright holders (it is not limited to the Regents of the University of California). In order to make the BSD-4-Clause that is specific to the Regents of the University of California distinct, SPDX has a separate BSD-4-Clause license, referred to as "BSD-4-Clause (University of California-Specific)"

COMMERCIAL COMPONENTS

Commercial Component tracking is essential in order to avoid audits, law suits, etc. The various elements (full product, runtimes, development environments, documentation, etc.) of commercial Components are typically licensed differently. As a result, the specific use case, including how each Component is licensed, is essential in determining compliance with your company's policy.

The following are some examples where the alignment of the commercial license and the Component's use case is critical:

- **Individually Licensed Content**: your company may have the rights to *use* a full copy of software licensed from a third-party (for instance Microsoft Visual Studio), but you may only have rights to *ship* or *host* a certain set of runtime components therein.

- **Expired Agreements**: unlike open source agreements, many commercial Component agreements may expire or must be affirmatively renewed. It is essential that any tracking system calendar the relevant notification and expiration dates.

- **Subsidiary-Specific Agreements**: many commercial Component agreements limit license grants to a specific subsidiary or acquired company.

- **Divested Products**: upon acquiring the assets of another company, many Component licenses do not automatically transfer to the new owner and must be explicitly assigned, at the licensor's discretion.

- **Product-Specific Agreements**: many commercial Component agreements grant rights to include Components only within a *specific* in-house product. The presence or absence of a particular Component must be considered in the context of the product it is included in.

- **Developer-Specific Agreements**: many commercial Component agreements grant unlimited distribution rights to a specific set of runtime components as long as the developer who creates your in-house product is specifically licensed for the *development tool* that is used to include the Component.

- **Location-Specific Agreements**: many commercial Component agreements limit distribution to a specific territory or site.

- **Identical Content Available under Multiple Licenses**: identical Components may be licensed under a variety of standard or custom commercial agreements, and may even be licensed differently depending on the OEM or support organization that supplied the Component. It is essential that the license agreement that is associated with a given Component be correctly identified, understood and checked against the engineering use case.

BEST PRACTICES

Scanning is an excellent tool for pre-acquisition diligence, and for periodic checking of an ongoing process, but it is not a sustainable or comprehensive process to protect your key investments – *engineering resources and intellectual property.*

A straightforward and low-cost approach is described in the Entente™ whitepaper titled, **A Practical Guide to Implementing an Inbound Technology Compliance Program**. This whitepaper describes the personnel, education, processes and toolsets required for an effective compliance program that leverages existing staff and resources.

The Entente method puts focus on the decisions that occur *prior* to including Components in your product. It is at this time–prior to inclusion–that license review, architecture review, competitive advantage, risk, etc. are most effectively and inexpensively analyzed and addressed. The Entente approach captures individually licensed components as they are evaluated, including license, use case, associated in-house products, source of technology, pricing, etc.

The Entente solution, IPCompass®, is pre-populated with thousands of the most commonly deployed open source and commercial component names and associates licenses. In addition, new Components and licenses specific to your company may be added via intuitive and contextual interfaces. Once a Component's characteristics have been entered into IPCompass, the information never has to be entered again and can be associated with any number of in-house products or bundles via simple drag-and-drop functionality.

The IPCompass method and solution are designed to deliver visibility to your company's entire *portfolio* of products and Components. It is this overall view that provides engineering leverage and cost savings. With a simple point-and-click, IPCompass will display open source and commercial Components that are packaged with any one or more of your in-house products. It is also easy to produce a bill of material that includes open source and commercial Components, showing what in-house products use a particular Component, license, etc.

Entente IPCompass is part of a practical and comprehensive open source and commercial compliance solution that assures that decisions regarding Component use are made before the Components are introduced into a product. Use of Entente's best practices and the IPCompass software solution provides visibility that has been repeatedly shown to reduce cost, reduce risk, highlight extensively used content, flag commercial and open source issues, and facilitate the full leveraging of third-party open source and commercial software components while carefully managing risk.