# Copyright Society of the USA
# Mid-Winter Meeting

## Feb 4-6, 2011
## Santa Fe, New Mexico

**Saturday, February 5**

10:45 am – 12:00 pm

**Copyright: What Is it Good for? Open Source, Social Media, and the Role of Copyright**

The ready availability of sophisticated technology to the general public has dramatically changed the IP landscape. In a world where participation and sharing are as valued (if not more so) as creation and ownership, concepts like author and copyright are increasingly being perceived by many users as quaint but dated concepts. Open source software and social media have come to thrive in this environment and touch almost every one of us daily, and content owners are coming to terms with how best to approach, embrace, or fight this phenomena.

This panel will explore what a new metaphysics of copyright might look like, in order to have increased relevance in this new world, from a variety of owner/user/aggregator/enabler perspectives.

# THIRD-PARTY CODE
## BEWARE THE TROJAN SOURCE!

BY KATHERINE C. SPELMAN AND BARBARA A. FREDERIKSEN-CROSS

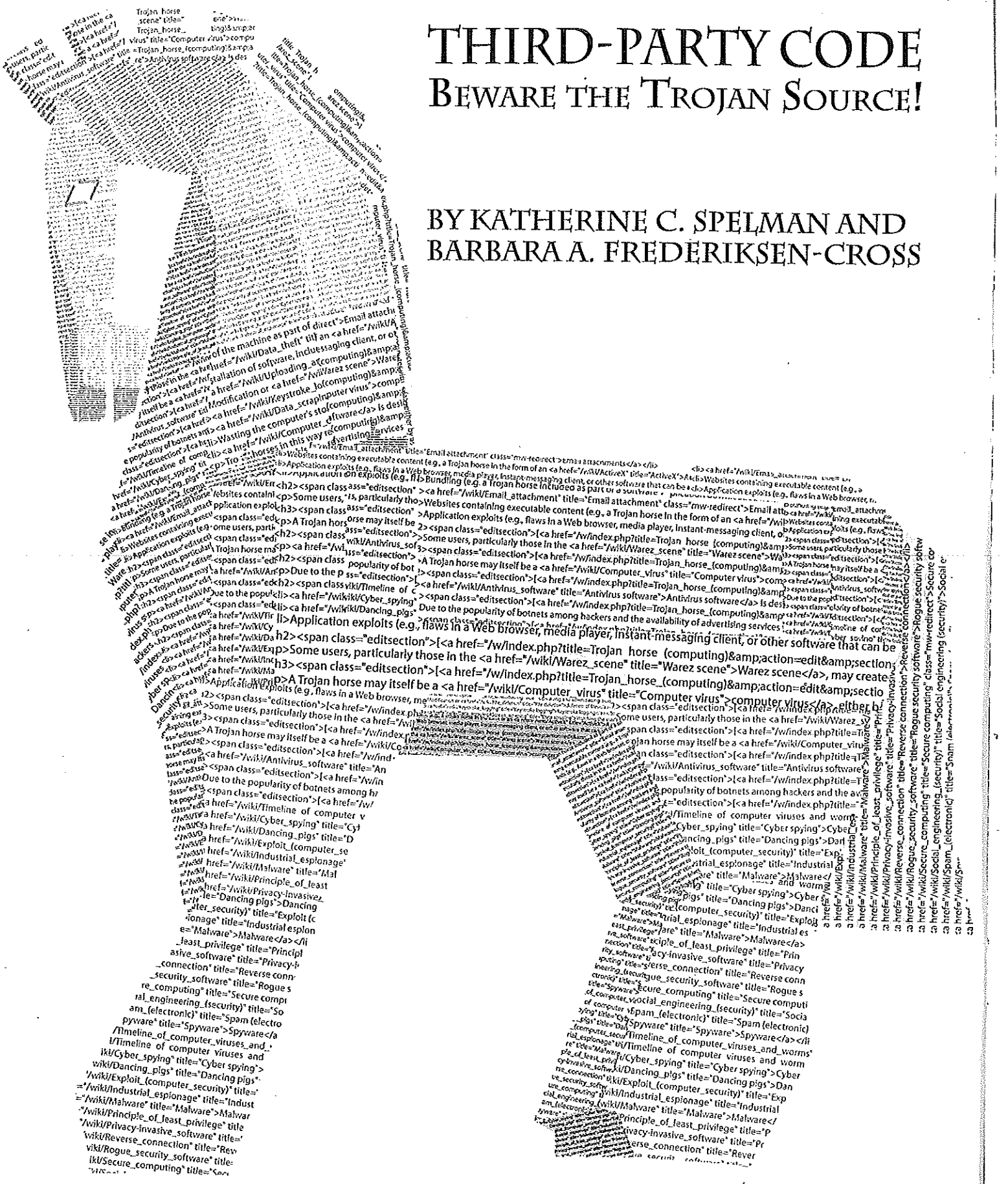



Illustration by Dan Mazanec

The use of open source components and other third-party materials is widespread in software development, but the use of such content introduces a variety of legal and security risks. This article identifies the most common pitfalls and advocates practical policies and procedures to help mitigate these risks.

## Introduction

Today, source code development is full, indeed thick, with third-party content. The reason is simple (and guided by the same principles that govern modern manufacturing): it doesn't make sense to reinvent the wheel for every automobile you build. Programmers are keenly attuned to this reality, and routinely reuse existing source code when developing new software. Cutting and pasting code from past development projects, Open Source repositories, third-party tools, or code found on the Internet accelerates new development. (*See* "What Is Open Source Software," *infra* at 16.) The use of these reusable solutions saves time, reduces debugging effort, and helps build more reliable code.

However, in addition to gaining these easy, quick advantages, the company that accepts such content also must accept new responsibilities. Code components obtained from Open Source repositories may be subject to the General Public License or other license restrictions requiring careful consideration before use.

Failure to manage the license and usage entanglements that accompany third-party content opens the door to litigation, trade secret loss, security vulnerabilities, and costly remediation efforts.

It would be unthinkable for a major company to begin constructing a new corporate headquarters without a clear knowledge of who owns the real property and what building codes or restrictions might apply to the construction site and land use, and without formal oversight and inspections during construction. In stark contrast, programmers who are developing the latest version of a company's flagship software often select, use, and propagate third-party materials without any formal review or oversight from management or legal counsel. This practice becomes fraught with peril when software developers fail to understand or comply with the licensing restrictions of the third-party material; fail to consider copyright, trade secret, or patent entanglements; fail to document the third-party use; or fail to consider security implications.

Each of the problems listed above can occur whether the third-party code is proprietary code or open source. We are all familiar with legal problems that arise when employees reuse code from past employers or consultants reuse source code that was initially written for a competitor. The potential for problems expands exponentially for open source, due in part to the fact that it is readily available to developers without any authorization from (or visibility to) management, and in part due to the proliferation of different licensing models that may apply to open source.

Open source licensing models may introduce significant risks and legal complexities for commercial organizations. (*See* "The 'Open Source' License," *infra* at 16.) Many open source and third-party programs contain prohibitions against commercial use that render them incompatible with software that is to be sold or licensed to customers. Open source licenses also may include "copyleft" requirements. Copyleft license terms require the source code for any product that embodies, uses, or derives from the third-party code subject to such license be made publicly available. This publication requirement would vitiate any trade secrets embodied in the new code and also may expose proprietary technology and techniques to the prying eyes of would-be competitors.

Use of third-party code (open source, public domain, licensed, or purchased) also may foster technical exposures such as security vulnerabilities. For example, if the security of an open source component is compromised, an individual who is aware of its security vulnerability may exploit the weakness to gain unauthorized access to the program function or to the computer system upon which it is running. Since the same open source routine may be incorporated into many different products, potential intruders may find a wide range of targets that all originate from the same vulnerability.

## Managing the Use of Third-Party Code

Management of open source and third-party content is best accomplished through prophylactic procedures, not reactive panic. Retrospective analysis is far more difficult and costly than proactive management of third-party content. The problems that stem from third-party code entanglements become more difficult to address as time passes. As comingled code evolves and changes, the actual origin of any particular code component becomes harder to ascertain. Modified code may be reused and spread to other products, making it harder to identify the scope of the original code's use and harder to contain or correct any damage or risk.

To proactively manage the use of third-party content, companies should establish a software pedigree program that introduces a series of "best practices" to code management. The essential elements of a software pedigree program are (1) carefully considered guidelines for using third-party materials, (2) a clearly defined review and approval process, (3) thorough documentation of third-party code use, and (4) the periodic performance of code audits to detect the presence of any undocumented third-party material.

Skills required to effectively manage third-party source code dictate the need for a core compliance team that includes participants from both legal and technical backgrounds, as well as input from individuals responsible for business development strategy. Participants with a legal background are needed to review license terms and assess legal risk factors relating to compliance and remediation efforts, trade secrecy protection, patentability, and the significance of ownership issues that might arise in the context of mergers,

**Katherine C. Spelman,** a partner with Cobalt LLC in Berkeley, California, specializes in copyright and trademark law. She also advises start-up companies, including those engaging in handheld computing devices and wireless technology, and has extensive experience in the beverage industry, offering expertise in wine issues as they relate to trademark and marketing law. She can be reached at kate@cobaltlaw.com. **Barbara A. Frederiksen-Cross** is a senior managing consultant, director, and forensic software analyst with Johnson-Laird, Inc., in Portland, Oregon. She can be reached at barb@jli.com.

acquisitions, or sale of the software asset. Participants with a technical background are necessary to assist in determining the scope and nature of existing third-party code usage. Once the pedigree program is in place, technical team members will help identify how proposed future code inclusions are to be used and whether the resultant product is for internal use or distribution to others. Technical team members also help determine whether third-party code use might imperil security and whether actions required for license compliance would expose sensitive or trade secret processes.

*Guidelines for Use of Third-Party Content*

Proactive management of open source and third-party content requires a published policy that defines where and how open source and third-party content can be used. This policy defines the criteria that will distinguish between approved and unapproved licenses and code origins. The scope of the management policy should address software developed by in-house developers, independent contractors, software vendors, and development partners. Where necessary, the management team should update contract language to address code developed by business partners, contractors, consultants, and outside developers.

Software pedigree documentation, which identifies the origin and use of all third-party components, should be formalized and produced as an artifact of new development. These records should be maintained and updated throughout the life cycle of the software to account for any changes to third-party content. Think of these records as self-insurance or as a metaphorical "get out of jail free" card that helps guard against expensive litigation.

*Review and Approval Process*

The policies governing third-party code management should include a review and approval process for the evaluation of third-party materials, the associated documentation identifying its origin, and license restrictions on its use.

The review process will typically include identification and assessment of the proposed usage, applicable license terms, legal considerations relating to licensing and IP ownership, and a review of any known security considerations with respect to the software or its use. All third-party code should be reviewed and approved prior to its incorporation into any shipping product or mission-critical software.

It is crucial to define a review process that is sensitive to your company's business needs, development models and deadlines, and risk tolerance. Although the review and approval processes may sound cumbersome, streamlining is easily achieved by the use of black lists, white lists, and grey lists. These lists will grow over time, and represent classifications of licenses and source origins into one of three categories.

- *The Black List* identifies materials that must never be used (e.g., materials developed for previous employers, materials whose source cannot be determined, and materials subject to unacceptable license restrictions);

## WHAT IS OPEN SOURCE SOFTWARE?

According to the Open Source Initiative Project, open source software distribution terms should comply with the following criteria.*

**1. Free Redistribution**

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

**2. Source Code**

The program must include source code and must allow distribution in source code, as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost, preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

**3. Derived Works**

The license must allow modifications and derived works and must allow them to be distributed under the same terms as the license of the original software.

**4. Integrity of the Author's Source Code**

The license may restrict source code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

**5. No Discrimination Against Persons or Groups**

The license must not discriminate against any person or group of persons.

**6. No Discrimination Against Fields of Endeavor**

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

**7. Distribution of License**

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

- *The White List* identifies preapproved materials (e.g., license terms and origin have already been accepted and scope of proposed new use is consistent with approved license terms); and
- *The Grey List* identifies materials that are candidates for use but require approval to assess license terms in the context of a specific proposed use.

*Record Keeping*

This review process includes maintaining a special repository for approved third-party code and associated licenses, as well as documentation to preserve a clear record of both the original third-party code base and the license under which it was obtained. Once code has been approved on the white or grey lists, it is also important to maintain records of where, in the larger context of a company's software, that particular code is actually used because the scope of use may grow over time.

The usage records should optimally provide information at the program or product level that identifies the specific version of third-party code used, and when it was implemented, the scope or context of its use (internal versus external, etc.). Additionally, the records should document which specific components incorporate the code and/or which components interact with it. If possible, all third-party code files should incorporate a comment block that documents the origin of the code and any restrictions that might apply to use of the code within a file.

Once approved, an original, unmodified copy of any open or third-party source code should be preserved. This copy will prove invaluable in the event patent, trade secret, or copyright disputes occur in the future. The complete license text, as well as any documentation relating to origin and use, also should be maintained, ideally in the same repository used for this source code.

Taken together, the repository of approved code and the records of its use form the basis for technical due diligence reporting. This information will be required in the context of mergers, acquisitions, and divestitures to properly characterize and identify your client's intellectual property.

*Education and Enforcement*

Once policies are in place, education all around of employees, contractors, and consultants is needed regarding their obligation to protect company assets. All software developers (in-house or consultants) should be trained in the actual mechanics of the review and approval process. As a part of this education, developers must be explicitly instructed to never reference or incorporate any code that was developed while working for past employers. Consider adding this policy to your company's internal code review process and providing periodic reminders about this policy to employees and contractors involved in software development.

Since software may be modified to respond to changing business needs, any proactive plan must include processes that apply to ongoing development and maintenance, as well as initial code development. It is extremely helpful to implement some form of regular, periodic audit or other policing process to identify potential problems that may arise during early implementation of the new policies. The policy also should include exception procedures to address existing code or other special circumstances.

**8. License Must Not Be Specific to a Product**

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

**9. License Must Not Restrict Other Software**

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

**10. License Must Be Technology-Neutral**

No provision of the license may be predicated on any individual technology or style of interface.

* *See, e.g.*, The Open Source Definition, http://www.opensource.org/docs/osd (last visited July 28, 2010).

**Audits for Legacy Code**

The procedures and guidelines described above are primarily directed to future development. Software developed before these management practices are in place must be subject to an initial audit to identify exposures relating to third-party content.

The code audit has two primary goals: first, to determine whether third-party material is present in a code base (a technical exercise), and second, to determine the ultimate origin and license associated with those third-party materials (both a technical and a legal question). The audit seeks to answer very basic questions: What third-party code are you using? Where did the third-party code come from? What license(s) pertain to its use?

The code audit itself does not remediate problem source code, but rather serves as an effective means to detect potential problems, providing useful information about the nature and scope of third-party code entanglements and identifying applicable licenses. A typical code audit is performed using a combination of automated tools and visual inspection to determine if third-party materials have been incorporated into proprietary code. A variety of tools and techniques may also be used, including sophisticated tools that can match your client's source code against vast databases of publicly available code bases. Even with the use of such sophisticated tools,

## THE "OPEN SOURCE" LICENSE

- In an "open source" or "public" license scenario, the rights holder usually makes the licensed work available to the general public without charge.
- The catch is that the user (the "licensee") must respect the ability of other downstream users to also use the work.
- Typically, open source licenses allow copyright owners to license certain uses of their works without charge, while reserving certain other rights.*

*David L. Applegate, *Railroaded Again?:* Jacobsen v. Katzer *and the Open Source Debate*, 1:6 LANDSLIDE 40, 41–45 (2009) (discussing open source licensing).

human analysis is still required to weed out false positives, catch false negatives, and, above all, judge the contextual relevance of a hit on third-party materials. Human judgment is also required in assessing the degree to which third-party code may present a risk to security, patentability, trade secrecy, or control of intellectual property.

A thorough search for third-party materials should also extend beyond source code libraries to include executables, documentation, and other types of files. In the author's experience, third-party materials are found, not just in the obvious source files or binary redistributables, but also in a variety of potentially unexpected locations such as font files, stored database queries, and online help functions. This list is by no means exhaustive. The critical point is that a comprehensive code audit must take the totality of the materials into account.

Regardless of the tools used, the product of a code audit should be a detailed record documenting every instance of third-party code, where it was found, what purpose it serves, and whether the code is used internally or distributed as a part of some product. These records should be sufficiently detailed so that every file is identified that includes third-party material. Where possible, the records should also preserve any information gleaned about the code's origin, owner, copyright, and license terms, as well as the specific third-party product or project from which the code was derived. This file-level metadata will prove invaluable in the event of challenge or litigation.

### *License Review*

Once third-party code inclusions have been identified, the next step is to identify the code origin and the applicable license conditions. Sometimes, comments within the source code itself will identify the specific governing license or the software product and owner. Armed with this information, licenses may be located via Internet searches, and the complete text may be downloadable. In other instances, the search for source code origins and the associated license text may present unexpected challenges. Sometimes the source code may have an obvious third-party attribution but may come from an unrelated company. This is not necessarily a contradiction, as it may reveal that a developer is recycling code written for prior clients and the attribution is residual detritus. In other cases, the software may reference companies (or individuals) whose assets have transferred as a result of mergers and acquisitions, or it may reference products that are no longer public.

The product of the license review should be a detailed record that includes the full text of each license and where and when it was obtained. As described in greater detail below, it will be important to maintain some enduring record that establishes which license(s) correlate to which programs or code portions. This correlated reference chart is essential because compliance evaluation will be done iteratively as software licenses change and may specify different terms for commercial and noncommercial usage.

### *Determining Compliance Status*

Once you have identified third-party code and any associated license(s), begin to assess compliance status and identify any related technical issues. This process has two primary components: a legal assessment of the current compliance status and an assessment of technical factors relating to the code and its use.

The focus of the legal assessment will be a review of license terms governing each third-party, source-code use to determine its compliance requirements, as well as a review of current practices to determine the current status with respect to those requirements. In some instances, multiple licenses may apply due to the collaborative nature of the code's development. Open source code may also be distributed under a choice of licenses, requiring extra research to determine which license applies.[1]

In addition to reviewing third-party software licenses, a compliance assessment will likely require the legal team members to review (and possibly revise) the following:

1. end-user license agreements or product packaging (to assess attribution compliance);
2. product revision and software download practices (to determine if required notices are shipped with each version of the product); and
3. Web- or phone-based download support (to review the process whereby GPL-based source code can be requested or delivered).

The technical assessment needs to provide program-specific information that will be considered during the legal assessment. For example, the same license may contain separate compliance requirements for commercial and noncommercial usage, necessitating a technical evaluation to determine the context in which the third-party code is used. In addition to addressing how third-party code is used, the technical assessment should also address whether code based on third-party software would pose a risk to security or trade secrets if license compliance required publication of the source code.

Careful recordkeeping during the review process is essential and will provide a foundation that minimizes the burden of future reviews. At a minimum, the recordkeeping should identify where compliance has already been achieved and the status of potential compliance issues requiring further research or action. Depending on the depth of the review, the review records may be useful to flag issues relating to intellectual property protection, Sarbanes-Oxley compliance, or

## FORENSIC CODE AUDITS: OBSERVATIONS

The authors' comments are distilled from hundreds of forensic code audits performed in response to litigation matters, technical due diligence for mergers and acquisitions, and internal code audits to proactively identify third-party code use. From that empirical experience, they offer these observations.

- Nearly every forensic code audit revealed evidence of undocumented third-party material in the code base.
- Automated analysis tools, while useful, are not substitutes for human analysis. The human brain remains the best judge of contextual relevance.
- Many companies' disclosures regarding the use of third-party materials prove to be incomplete, suggesting that many, if not most, companies have yet both to understand this vulnerability and to implement effective pedigree tracking practices.
- Developer education is crucial, as multimillion-dollar litigation may be launched in response to the actions of a single, uninformed developer.
- In acquisition scenarios, deals have been terminated after unexpected discovery of third-party materials in a seller's code base (often contradicting the seller's disclosures). A proactive audit can help identify potential problems.
- Proactive audits can have big payoffs in mergers and acquisitions. Proper identification, disclosure, and management of third-party content minimize the very real risk that sophisticated buyers will force closing-day discounts when the seller cannot comply with technical due diligence disclosures.
- The validity and enforceability of copyright registrations depend upon appropriate identification of pre-existing works, including open source and third-party components.

Available online tools that may help you to spot, analyze, and remove vulnerable programs include Top 100 Network Security Tools, http://sectools.org/ (last visited July 28, 2010) and Open Source Code Analyzers in Java, http://java-source.net/open-source/code-analyzers (last visited July 28, 2010).

Companies such as Palamida, http://www.palamida.com/ (last visited Oct. 21, 2010); OSRM, http://www.osriskmanagement.com/ (last visited Oct. 21, 2010); and Black Duck, http://www.blackducksoftware.com/ (last visited Oct. 21, 2010), offer tools that compare a body of source code against a repository of code fingerprints generated from a variety of publicly available software projects.

similar categories of concern.

Generally speaking, the compliance recordkeeping needs to be program-centric, rather than license-centric. The reason is that the same piece of third-party code may be reused multiple times in different contexts (e.g., used internally and also used as a part of a distributed product) and may be subject to different license terms depending on the context of its use.

### *Remediation*

The specific steps taken to address compliance related to third-party code use (a.k.a. "remediation") must consider both the needs of the business and the terms of the license. Where remediation is necessary, it will generally require:

1. modification of existing documents and processes to comply with the license terms,
2. removal or replacement of third-party code, and/or
3. attempts to negotiate for more acceptable terms.[2]

Unfortunately, there is no general rule to guide retrospective remediation; every license and code use will need to be addressed on a case-by-case basis. In some cases, legal factors—such as license terms that prohibit all commercial use—may drive this decision process. In other instances, technical factors—such as the importance of the third-party code, the cost to replace it, the degree to which the code has been modified for integration, and resource or time constraints—may limit effective remediation. ■

### Endnotes

1. Various types of license agreements exist. However, the General Public License is used by many of today's leading software producers and companies within the electronics industry. More information about the General Public License can be found at http://www.gnu.org/licenses/gpl.html. For an analysis of the second versus the first version of the General Public License, see Terry J. Ilardi, *GNU and Improved?: The Newest Version of the General Public License for Software—What to Tell Your Client*, 1:3 LANDSLIDE 38 (2009) (discussing the latest version of the General Public License).

2. This form of remediation may be successful if the negotiation occurs before the code is actually used. Its likelihood of success diminishes once the third-party code is incorporated into live software.