GUIDE

Evaluating Open Source Content in Software Assets in M&A Due Diligence

If there's open source risk, pre-close is the time to discover it.

Most companies involved with technology M&A understand the importance of open source risks in software. Today's software contains significant amounts of open source, on average more than 50%, according to a 2018 Synopsys study.[1] Because few companies manage their developers' use of open source effectively, most codebases contain open source components that carry license or security risks. It has become the norm for acquirers to raise open source questions as part of technical and legal due diligence.

There are several ways to assess and manage open source risk in a transaction, with some more effective than others. Similarly, there are several approaches to open source due diligence and varying degrees of depth of analysis.

Reps and warranties

One approach is to cover the risk with reps and warranties—that is, to have the seller represent that they're in compliance with all open source licenses (beware of knowledge qualifiers). However, as most acquirers realize, it's much better to know of issues before close and have the comfort of reps and warranties for worst-case protection after close. It's also better to ascertain the open source content and associated risks as part of up-front due diligence. If there's open source risk, pre-close is the time to discover it, so addressing risk can be worked into terms and integration plans.

Asking the target vs. employing a third party

The simplest approach to determining what open source is in a codebase is to ask the target. It's good practice to do so in any case, but many targets have difficulty producing a full and accurate inventory (also known as a "bill of materials" or BoM) of open source in their code. Albeit not rocket science, managing and tracking open source requires an explicit policy, processes, technology, education, and enforcement, which most companies lack.

To be sure of what's in the code requires actual analysis of the code. Most targets are reluctant to grant extensive code access to potential acquirers. Some targets have tools to analyze their own code and be equipped to provide information. However, most open source tools require human involvement and judgment. The skills and experience of the user matter a lot, so savvy acquirers will question the accuracy of results produced by a target. Given all those reasons, acquirers often involve independent third parties to do analysis.

The skills and tools available to third parties vary. Some acquirers, particularly PE firms, outsource all or much of their technical due diligence. Firms that engage primarily in broad technical due diligence typically lack specialized open source identification tools and specialized open source skills. Their personnel are seasoned software professionals with broad experience and expertise in different software technologies. They're skilled in assessing the processes, architectures, and personnel employed by the target, but not necessarily open source risk. Most have a reasonable understanding of open source, but not the depth of experience an expert would have.

Analysis techniques

There are a number of analysis techniques that can reveal open source in a codebase. Each has strengths and weaknesses; none is sufficient by itself. A comprehensive approach employs multiple techniques.

The most basic single technique is string search. Searching for a few keywords, such as "copyright" or "license," leads an auditor to lines in the source code that may be near license text or a reference to an open source license. However, these general searches can be quite inefficient, and their success depends on the open source experience of the auditor, who must be able to find, recognize, and log open source license text.

1 Synopsys, 2018 Open Source Security and Risk Analysis, 2018.

A comprehensive approach to open source analysis employs multiple techniques.

String searches alone miss many instances of open source and provide limited information about the open source. For example:

- **Binaries.** In modern software, open source components are commonly included in binary form as libraries. Most often, string searches won't find strings in binaries, and when they do, the results are generally without context. One might learn that the word "copyright" is buried inside the binary, but not have any further information about it.
- **Dependencies.** Modern software includes callouts to other files, which may not be present. Those files may be open source, but the string search won't "see" those files so won't help to identify them as such.
- **Source code.** The idea of string search is to find clues to or actual license text in the comments in source code. Well-documented source code should include license information, but much code does not. An open source component may be distributed with a ReadMe file in the main directory, but with no indication of license information in individual files. Even if the license information is in the files, a developer may have stripped out the "useless" header containing license information or may have cut-and-pasted a few hundred lines of code from the middle of the file with no license text included. In all these cases, string search can fail to identify even source code as open source.

Even when auditors find license information, they may be able to identify only the license for the component, not the specific open source component. With some informed sleuthing, experienced open source auditors may find clues that allow them to guess the identity of a component, but often the open source component remains a mystery. In those cases, the information still can be useful for checking license compliance, but not for creating a comprehensive software BoM. Without that component identity, an auditor could not, for example, surface known security vulnerabilities associated with the component. In these post-Equifax days, it's critical to be mindful of the security of open source in addition to resolving any license issues.

All the abovementioned examples can be identified and analyzed with tools designed for discovering open source via a comprehensive set of techniques. Black Duck audits employ the techniques described earlier, as well as more sophisticated, complex, and comprehensive techniques. Because most discovery techniques involve some matching to known open source, they depend on a database, which must be extensive to be effective. For example, the Black Duck KnowledgeBase™ contains source and binary information for millions of open source components. Obviously, the tools used to compare a large codebase against such a vast database must be sophisticated and highly optimized and are thus unavailable to many due diligence consultants.

Conclusion

In the high-risk world of M&A, involving open source–specialized auditors with experience and sophisticated tools is the appropriate approach whenever a target's software assets are a significant part of the valuation. The Black Duck audit services group, the first name in open source auditing, employs the best tools and has assembled the largest team of auditors specialized in identifying open source code.

How a Black Duck open source software audit works

Black Duck audit customers are generally either acquirers looking to audit the code of their target or companies wanting to audit their own code in anticipation of being acquired. If the context is an M&A transaction, there's often significant time pressure. While we regularly respond to last-minute requests, customers save more time, money, and headaches the earlier they decide to get an open source audit and establish the scope of work involved.

The Black Duck audit services group works with the "code owner," often a third party in an M&A transaction, to get a high-level view of the composition and complexity of the codebase and its architecture. In great part, the scope of work is driven by the number of files and the prevalence of open source components in the technologies used. For example, JavaScript tends to be full of open source files, whereas a typical C++ codebase contains much less open source.

We work with customers to find the best balance between scope of work, schedule, and cost to perform an audit that best fits individual needs and budgets.

Before the audit commences, there's typically an NDA with the third-party code owner when an acquirer engages Black Duck audit services. Ultimately the scope is agreed to through a statement of work that describes work to be done, codebase, schedule, and fixed price. We work with customers to find the best balance between scope of work, schedule, and cost to perform an audit that best fits individual needs and budgets.

We can perform open source code audits either on or off the customer site. Code owners are usually comfortable with securely uploading code to our servers. The benefits of this approach include minimizing costs and time and maximizing flexibility (if, for example, the job expands and more resources are necessary). However, in some cases, code owners require the audit team to come on-site.

As soon as they have access to the code, the Black Duck audit team begins the identification process. This process is semiautomated, meaning that identification relies on a set of tools as well as the expertise of auditors. For comprehensiveness, our tools employ a variety of techniques to ferret out unknown open source. In many cases, the tools definitively identify open source components, but sometimes, owing to limited information in the code, they just provide clues for expert auditors to chase down.

The result of this identification process is a comprehensive bill of materials (BoM), essentially a list of the open source components in the codebase. The BoM is the foundation for identifying open source risks. Only by knowing what's in the code can you know the associated risks.

We enumerate three types of issues—legal risks, security risks, and operational factors:

- **Legal risks** are primarily the result of using a component in a way that conflicts with the terms of the open source license.
- **Security risks** stem from using components that have known security vulnerabilities.
- **Operational factors** come with components that are out-of-date or inactive.

In addition to identifying issues, the audit team provides red/yellow/green rankings for each to help customers with prioritization. We use the BoM and risk rankings to create a report that we deliver to our customer via a secure portal. Via the portal, customers can review the report or download it and share it with associates.

Post-audit

We recommend a post-audit review call, during which our auditor walks through the report and answers questions about how it was generated. Ideally, the call includes customer staff or advisors who can interpret the implications of the risks in light of the customer's unique situation. It almost always makes sense to have legal counsel involved, particularly counsel familiar with open source licensing. And it's a good idea to have people on the call familiar with the architecture of the codebase and who understand cyber security.

The extent and severity of the issues identified vary, but there's almost always some "cleanup" required. In the case of an acquisition, the closing may wait for remediation, or the parties may agree to take care of things post-close. In some scenarios, customers want to verify that all identified issues have been addressed. After remediation, we can perform a verification scan and provide a new (now presumably clean) report.

The Synopsys difference

Synopsys Software Integrity Group provides integrated solutions that transform the way development teams build and deliver software, accelerating innovation while addressing business risk. Our industry-leading portfolio of software security products and services is the most comprehensive in the world and interoperates with third-party and open source tools, allowing organizations to leverage existing investments to build the security program that's best for them. Only Synopsys offers everything you need to build trust in your software.

For more information, go to www.synopsys.com/software.

Synopsys, Inc.
690 E Middlefield Road
Mountain View, CA 94043 USA

Contact us:
U.S. Sales: 800.873.8193
International Sales: +1 415.321.5237
Email: sig-info@synopsys.com