

WHITE PAPER

2026 OPEN SOURCE RISK IN M&A BY THE NUMBERS

AUDIT DATA INSIGHTS

The findings in this paper are derived from the anonymized data produced by the Black Duck Audit team while evaluating commercial and proprietary codebases, primarily in support of merger and acquisition (M&A) transactions, between November 2024 and October 2025. (For simplicity's sake, we refer to the overall period as "2025" in this paper.)

Best practices for firms involved in M&A transactions include audits of the target's code whenever software is a significant part of the value of a deal. Because targets generally don't share code with potential acquirers, expert third parties that perform multiple types of analyses on the code can help purchasers better understand the technology and capabilities they're buying and identify potential legal, security, and quality issues.

Open source is effectively universal in commercial software. Only 2% of the codebases we audited in 2025 contained no open source components—and even that figure typically represents specialized legacy systems rather than new development. For practical purposes, if you're building software today, you're building with open source. And lots of it. It's been almost a decade since we first noted that code in targets' applications was more open source than proprietary code.

Open source components and libraries form the backbone of nearly every application in every industry. The reasons are straightforward: Using open source speeds development, drives innovation, and lowers costs, all critical in today's agile software world. Developers know this, and the AI tools many are leveraging today were trained on and pull from millions of publicly available open source repos. The Black Duck Audit team conducts open source audits—as well as analysis of software security and quality—on thousands of codebases (the code and associated libraries that make up an application or service) each year. Our audits provide customers with comprehensive, up-to-date Software Bills of Materials of the open source, third-party code, web services, and APIs used in their applications, and to enable a view into risks associated with the components.

OPEN SOURCE RISKS

The 2026 "Open Source Security and Risk Analysis" (OSSRA) report presents detailed analysis from an examination of the anonymized data from about 1,000 codebases audited as part of hundreds of M&A transactions in 2025. Industries represented in the report include automotive, big data, cybersecurity, enterprise software, financial services, healthcare, the Internet of Things, manufacturing, and mobile apps.

The OSSRA report presents data on codebases. As this paper focuses on M&A transactions, data is presented by transaction, meaning the software being acquired via the transaction (which usually involves more than one codebase).

Acquirers in M&A deals want to understand what risks may be associated with the software they're acquiring. For the open source elements of a target's code, the concerns are with the licensing, security, and quality of the specific components used in that software. The 2025 audits found open source in 98% of codebases and as part of 100% of transactions with an average of 3,550 open source components per transaction. Given the time-to-market, cost savings, and development advantages of leveraging open source components, it's no surprise that companies continue to rely heavily on open source as part of their software development process. But the large number of discrete components speaks to the challenge of tracking it all.

The vulnerabilities and license compliance issues discovered in the codebases were almost as pervasive as open source itself. In 2025, we found components with license conflicts in 94% of transactions, up a bit from the previous year. Commercial codebases with license conflicts can create legal exposure and compliance failures.

Modern applications can accumulate hundreds of transitive dependencies, resulting in license obligations that are extremely difficult to track manually. Identifying them requires automated tools capable of deep code analysis well beyond what manifest scanning can provide.

94% of transactions included components with license conflicts.

"Snippets"—lines of code that have been copied and pasted into source code—are quite frequently found by the Black Duck Audit team.

In our audits, 84% of open source components were identified through automated detection methods—package manager analysis and manifest file scanning. But the 16% requiring deeper analysis tells an important story. This code entered the codebase outside of standard package management: vendor dependencies copied directly into repositories, code snippets pulled from Stack Overflow or generated by AI assistants, and binary components included without source. These components are invisible to tools that rely solely on manifest files.

LICENSING RISK AT A GLANCE

The most common licensing conflict in target code involved the Creative Commons Attribution ShareAlike (CC-SA) 3.0 license, which was found in 54% of transactions. As noted above, snippets are often taken from the popular blog Stack Overflow, which automatically licenses all publicly accessible user contributions under Creative Commons Attribution ShareAlike. Unfortunately, the blanket license also covers code snippets posted on the site. We say “unfortunately” because Creative Commons licenses are not intended for software, with Creative Commons explicit about this in its FAQ: “We recommend against using Creative Commons licenses for software.” Misapplying such a license adds to the complexity of legal analysis.

The CC-SA license is similar to the GNU Public License (GPL)—a license that also turns up in frequently audits—in that any work derived from work using this type of license must be licensed under the same terms. And that can become a concern from a legal standpoint to companies that distribute software.

License obligations also apply to code snippets, no matter how small. And it’s important to note that snippets are also increasingly being injected into code by AI coding assistants. The 2025 “GitHub Copilot Statistics & Adoption Trends” report reveals that developers accept an average of 30% of AI-suggested snippets, though this rate fluctuates based on developer experience.

SECURITY RISKS AT A GLANCE

Of course, licenses are not the only concern with using open source software. Unpatched software vulnerabilities are one of the biggest cyber threats organizations face, and unpatched open source components in software add to security risk. Ninety-seven percent of transactions included unpatched, vulnerable open source, with a mean of 786 vulnerabilities per transaction. Seventy-eight percent of codebases also contained open source with at least one high-risk vulnerability. High-risk vulnerabilities are those that have been actively exploited and already have either a documented proof-of-concept exploit or classification as a remote code execution vulnerability.

97% of transactions included vulnerable open source.

Our audits showed that the Linux Kernel was the component that contained the most vulnerabilities. The Linux Kernel team became a CVE Numbering Authority (CNA) in early 2024, resulting in a dramatic increase in kernel-related CVE assignments—from 290 in 2023 to over 3,500 in 2024. For any codebase with Linux Kernel dependencies, this single change added hundreds of vulnerabilities to the count without any change to the underlying code. It’s also worth noting that only 2% of the codebases we scanned included the Linux Kernel, meaning that only a handful of very large codebases skewed the overall vulnerability results.

jQuery was the #2 component containing vulnerabilities. Sixty-four percent of audited transactions contained the jQuery component. This is not to say that jQuery is inherently insecure. In fact, it is a well-maintained open source library with a large community of users, developers, and maintainers. But according to the audits, jQuery was the second-most-likely component to have vulnerabilities, even though all the jQuery vulnerabilities listed in the 2026 OSSRA report have available patches. To reiterate, this is not an indictment of the quality of jQuery, but rather of the processes that companies use to monitor and patch vulnerabilities. Issues we found were a result of the component not being kept up-to-date. It is important for users of jQuery—and indeed users of all open source—to be aware of the potential security risks associated with the use of older versions of software, and to design processes to stay current and mitigate those risks.

VERSION RISKS AT A GLANCE

In 93% of codebases, the software included components that had no new development activity in the last two years.

Of the customer transactions conducted by the Black Duck Audit team in 2025, 93% of codebases audited included components that had no new development activity in the last two years, an issue that appeared in nearly every transaction. Similarly, nearly every transaction and 92% of codebases had open source more than four years out-of-date, that is, using open source with newer versions available—usually with many newer versions available.

Besides contributing to security risk, getting too far behind in versioning means that software is falling behind functionally and is becoming more difficult to update. That is because the more versions that need to be leapfrogged when upgrading a component, the harder it is to integrate the latest version. Thus, falling behind can be costly and risky when an upgrade becomes necessary to eliminate a vulnerability or for any other reason. That creates a “double-whammy” effect of older components: they are both more likely to be vulnerable and harder to fix when they are.

What should companies be doing to address these issues? Best practices in the use of open source software require development teams to know which components and associated licenses are in their code and what obligations result from their use of that open source. It also requires monitoring for newly discovered vulnerabilities and keeping up-to-date with the latest versions. Fundamentally, this requires tracking what open source components are in a codebase. However, doing so manually can be a near-impossible task for any organization.

Customer audit engagements in 2025 conducted by the Black Duck team found a huge number of open source components (over 1,000 components in an average codebase), each with versions, vulnerabilities, and licenses that need to be tracked.

This volume of components suggests the importance—if not absolute necessity—of automated open source management. At that scale, a company cannot rely on manual processes, and an acquirer cannot assume that a target is comprehensively tracking the open source it uses.

The reality is that many companies, particularly smaller ones, don't have the necessary processes and tools to manage their developers' use of open source; it's simply not a high priority to the typical startup. Although we've seen improvement since we started reporting 11 years ago, it is still the case that most targets are unable to produce an open source Software Bill of Materials easily and routinely. When they do, it is rarely accurate and never complete. And even for such open source as companies are aware of, they don't tend to track security vulnerabilities, as evidenced in the statistics in this paper.

SOFTWARE COMPOSITION ANALYSIS FOR M&A DUE DILIGENCE

Knowing what open source code is in a company's codebase is crucial for properly managing its use and reuse, ensuring compliance with software licenses, and staying on top of patching vulnerabilities—all essential to containing business risk. From an M&A perspective, a code audit enables a buyer to understand risks in the software that could affect the value of the intellectual property and the remediation required to address those risks, as well as to plan out a roadmap going forward. Savvy sellers may employ an audit proactively to avoid surprises in due diligence, particularly given the amount of unknown open source in a typical company's code.

This is why an open source audit can be invaluable for companies wanting a better understanding of the code's composition. Using techniques known as software composition analysis (SCA) and a range of sophisticated tools, expert auditors comprehensively identify the open source components in a codebase and flag legal compliance issues related to those components, prioritizing issues based on their severity. The audit identifies known security vulnerabilities that affect the open source components, as well as information such as versions, available patches, and the state of a component's development activity. It also provides indications as to the sophistication of a target's software development processes. Open source is so ubiquitous today that if a company isn't managing that part of software development well, it raises questions as to how well it is managing other aspects.

If you're on the buy side of a tech M&A transaction, an open source audit should absolutely be part of the software due diligence process. Acquirers need to identify problematic open source in the target's code before the transaction terms and integration plans are set, and a trusted third-party audit is the best way to get a deep, comprehensive view. Prospective sellers should prepare for questions about the composition of their code and how well they have managed open source security and license risk. Absent good open source management processes, proactive sellers can prepare for an acquisition by having their software audited in advance.

OPEN SOURCE LICENSE COMPLIANCE

Like all software, open source components are governed by licenses that vary in terms of rights, obligations, and restrictions. Failure to comply with open source licenses can put businesses at risk of litigation and could compromise their intellectual property. Generally, as part of a definitive agreement, sellers need to represent that they have the rights to any software they are using (and there may be even more explicit representations and warranties regarding open source). Yet few sellers are totally “clean” when it comes to open source license compliance.

Organizations can manage and comply with license requirements only if they can identify the open source components and confirm that the use of those components is consistent with the terms of the applicable license. Just as with security vulnerabilities, it’s impossible to manage license compliance risk without identifying all the components in the software.

Based on Black Duck audit data, investors in all software verticals should be concerned about open source licensing and the potential risk of litigation or threat to their intellectual property rights due to failure to comply with an open source license. Black Duck analyses indicate that the 20 most popular licenses cover approximately 98% of the open source in use. But whether the software in question uses one of those popular licenses or some variant, the license matters.

In 69% of transactions, auditors found open source with customized licenses, or components freely available on the internet but with no discernible license at all.

License risk arises when software includes components with licenses that conflict with the way the overall codebase is licensed. For example, the GPL is an extremely common license that often governs components in commercial software and requires distributors to make source code available. But commercial software vendors typically do not offer to provide source code or complete source code, which creates a conflict with that license.

Sometimes an open source component has what Black Duck refers to as a “custom license” in which the developer created their own licensing language or added language to create a variant of a standard license. Customized open source licenses might place undesirable requirements or limitations on the licensee and will often require legal evaluation for possible IP issues or other implications. For example, the JSON license is based on the permissive MIT license, but the JSON license adds the distinction that “the software shall be used for good, not evil.” The ambiguity of this statement leaves its meaning up to interpretation, posing a particular concern in M&A scenarios where acquirers and their counsel need to weigh any legal risk. And there are many examples of developers being similarly “creative” with licenses.

Black Duck audits conducted in 2025 found that 69% of transactions included open source with customized licenses or no license at all. If third-party code is used without a license, this raises legal concerns. In the U.S. and many other jurisdictions, creative work—including software—is placed under exclusive copyright by default. Unless there’s a license from the copyright holder that grants permission, no other party can use, copy, distribute, or modify the software without the risk of litigation.

100% of the transactions in 12 of the 17 industry sectors had open source license conflicts, with the lowest being 71% in one industry sector.

Permissive and Reciprocal Licenses

Open source licenses fall on a spectrum from permissive to reciprocal. Permissive licenses place minimal obligations on companies that redistribute the associated software. By contrast, reciprocal (also known as “copyleft”) licenses require the licensee to make any improvements or enhancements available to the public under similar terms. In some cases, the entirety of the work that incorporates the licensed software, even a small portion of it, may fall under reciprocal obligation.

Permissive Licenses

Permissive open source licenses generally require only that the licensee attribute the original portions of the licensed code to the original developers, both in the code and in documentation. Of course, to provide such an attribution, the licensee must be aware that they’re using the licensed code, so it is no surprise that most targets are not fully compliant with permissive licenses. While this is

generally considered a lesser risk than, say, distributing GPL-licensed code, it is still an issue that most acquirers will want to address in their plans to meet their own corporate standards, typically stricter than those of most targets.

Reciprocal Licenses

On the other end of the spectrum, codebases containing reciprocal licenses are quite problematic for an acquirer. Many of these licenses require associated code to be made available to the public under the same license. A licensee that violates a reciprocal license could be at risk of litigation and may be required to disclose all the source code of the application. Such issues often come to light when a licensee is acquired in an M&A transaction. Acquiring companies will want to remediate either before or after close.

IP COMPLIANCE RISK INTRODUCED BY AI CODING TOOLS

Arising with the use of Generative AI-powered coding tools are questions around ownership, copyright, and licensing of the generated code. For example, a class-action lawsuit filed against GitHub, Microsoft, and OpenAI claims that GitHub Copilot—a cloud-based AI tool that offers developers autocomplete-style suggestions as they code—violates both copyright law and software licensing requirements. The lawsuit further claims that the code suggested by Copilot seems to, at times, copy/paste licensed code without attribution, copyright notice, or adherence to the original licensing terms. Microsoft offers some level of indemnification when the tool is used under certain guidelines, but user companies still need to be wary as many other such coding tools do not provide any protection.

The Copilot case highlights the legal complexities surrounding the use of AI-generated code. For software developers, refraining from using AI-assisted coding tools until the issue is resolved by legal or government decision is obviously the safest way to avoid an action for license or copyright violations, but the reality is that a growing number of developers are using such tools and will continue to do so. An open source audit as part of software due diligence will highlight instances where an AI has “copied” code from open source projects and associated risks, just as with human-written code.

OPEN SOURCE MAINTENANCE

Large organizations may manage hundreds to thousands of software assets, ranging from mobile apps to cloud-based systems to legacy systems running on premises. That software is typically a mix of commercial off-the-shelf packages and custom-built codebases, both of which contain volumes of open source components.

As noted earlier, the Black Duck Audit team found open source in 98% of codebases in 2025. Here’s the reality: If an organization builds, sells, or simply uses software, it’s safe to assume that the software contains open source.

In 97% of transactions in which we were involved, we found unpatched open source vulnerabilities, with a mean of 786 vulnerabilities uncovered per transaction. Seventy-eight percent of the codebases in those transactions had at least one high-risk vulnerability. “High-risk” indicates that a vulnerability has been actively exploited, has documented proof-of-concept exploits, or has been classified as a remote code execution vulnerability.

Open source projects usually issue small updates at a much higher frequency than the average commercial software vendor. When these updates contain security updates, companies need to have a strategy to adopt them rapidly. But because open source updates need to be “pulled” by users, many companies consuming open source components don’t apply the patches they need, exposing their business to the risk of attack and their applications to potential exploits. This is understandable—these companies typically aren’t even aware of the outdated components in their code, let alone looking for patches.

The data indicates that development teams may be struggling with the dynamic nature of open source security risk, especially with the increase in open source use. An open source component with no known vulnerabilities doesn’t necessarily stay that way a year, month—sometimes not even a week—later. So unlike with licenses that don’t change with great frequency, up front vetting is insufficient to ensure code security.

The struggle is illustrated by organizations being startlingly behind in using the latest version of any given open source component. Nearly 100% of our 2025 M&A transactions included open source components more than four years out-of-date and components that were not the current version—exposing the codebases to security risks and other issues. Firms acquiring companies, undergoing mergers, entering joint ventures, or even managing underlying technology supply chains must also consider the risks they are potentially inheriting.

SUMMARY

The thousands of audits conducted by the Black Duck Audit team have consistently revealed that almost every codebase contains open source code. As documented in this paper, 98% of codebases audited in 2025 included open source components.

The Black Duck Audit team generally audits codebases from software-heavy companies, as opposed to enterprises that use software to support their business. The primary value of software companies is their proprietary code. The ratio of open source to total code in their codebases, while still quite high, is eclipsed by how much open source is used in large enterprises. The figures cited by analysts such as Forrester, which generally look at enterprise IT groups for their reports, consistently find that open source often comprises up to 90% of new codebases, and that over 90% of IT organizations use open source software in mission-critical workloads.

With the magnitude of open source usage comes risk, due primarily to organizations lacking the needed tools and processes to recognize what—and how much—open source is in their internal and public-facing applications.

Failure to comply with open source licenses can put businesses at significant risk of litigation and jeopardize their ownership rights to their software. Perhaps more importantly, especially in the context of M&A transactions, failure to comply with open source licenses more likely to result in demands for additional acquirer protections, pre-close remediation, escrows, purchase price reductions, or deal delay. Ninety-four percent of transactions audited in 2025 involved components with license conflicts. Sixty-nine percent had open source with customized licenses or no discernible license at all. In other words, license issues are pervasive in the M&A deals we see, and they are potentially costly. At minimum, acquirers need to be mindful of these risks and reflect remediation in their post-close plans.

By identifying open source code and third-party components and licenses, an open source audit can alert your firm to potential legal and security issues in an M&A transaction. With an open source audit, you can also

- Avoid surprises
- Mitigate legal exposure
- Understand risks that may affect software asset values
- Resolve potential issues before they affect the transaction
- Build appropriate protections into the deal terms
- Plan integration and remediation

ABOUT BLACK DUCK

Black Duck[®] meets the board-level risks of modern software with True Scale Application Security, ensuring uncompromised trust in software for the regulated, AI-powered world. Only Black Duck solutions free organizations from tradeoffs between speed, accuracy, and compliance at scale while eliminating security, regulatory, and licensing risks. Whether in the cloud or on premises, Black Duck is the only choice for securing mission-critical software everywhere code happens. With Black Duck, security leaders can make smarter decisions and unleash business innovation with confidence. Learn more at www.blackduck.com.