

GUIDE

## **Better. Faster. Stronger.**

Complementing Static Analysis With  
Software Composition Analysis for  
Open Source Management

# Table of contents

<b>Introduction</b> .....	<b>2</b>
<b>Looking at unmanaged open source risk</b> .....	<b>2</b>
License risk of unmanaged open source.....	2
Quality and maintenance risks of unmanaged open source .....	2
Security risk of unmanaged open source .....	3
<b>Managing your open source use: The DIY approach</b> .....	<b>3</b>
Step 1: Inventory your open source .....	3
Step 2: Control your open source .....	4
Step 3: Ensure license compliance .....	4
Step 4: Ensure code quality .....	4
Step 5: Map open source to known vulnerabilities .....	5
Step 6: Manage risk and monitor for new threats .....	5
<b>SCA: A better, faster, stronger path to open source management</b> .....	<b>6</b>
Better: Get a complete view of the open source in your codebase .....	6
Better: Eliminate open source license noncompliance.....	6
Faster: Continuously monitor the quality and security of the open source you use.....	7
Stronger: Combine Black Duck SCA with Coverity SAST .....	7

Over 60% of the code in an average application is composed of open source components.

Do you know whether the open source components in your applications use permissive or viral licenses?

Do you know whether the open source components in your applications are abandoned?

## Introduction

Static application security testing (SAST) tools, such as Coverity® [SAST](#), are critical for uncovering and eliminating issues in proprietary code early in the software development life cycle (SDLC) by scanning code for flaws while that code is in a nonrunning (i.e., static) state.

However, SAST isn't effective in finding third-party open source software vulnerabilities (CVEs) or identifying open source license types or versions. Open source is an essential component of application development today, with over 60% of the code in an average application composed of open source components.<sup>1</sup> Adding a [software composition analysis](#) (SCA) tool, such as Black Duck® SCA, is as imperative to your software development strategy as using SAST to test the code your developers write.

This guide looks at some processes you can put in place to manage your open source use and strategies to manage open source risk. It also highlights the challenges you'll face along the way and recommends an SCA approach to address or avoid those issues.

## Looking at unmanaged open source risk

### License risk of unmanaged open source

An [open source license](#) is a type of license that allows the source code to be used, modified, or shared under defined terms and conditions. Black Duck SCA scans indicate that the 20 most popular licenses cover approximately 98% of the open source in use. Whether a particular license is one of these popular licenses, a variant license, or one of the thousands of less commonly seen licenses, it almost always has specific restrictions and obligations that users must comply with. For example, a "permissive" license—the most basic type of open source license—allows you to do whatever you want with the code as long as you acknowledge the authors of the code and follow other obligations such as redistribution and documentation requirements. A viral or "copyleft" license adds further requirements to the permissive license. For example, if you distribute binaries, you must make the source code for those binaries available. The source code must be available under the same copyleft terms under which you originally got the code; in other words, you must give it away for free. You can't place additional restrictions on the licensee's exercise of the license.

Failure to comply with open source licenses can put you at significant risk of litigation and compromise of intellectual property (IP). In their experience performing code audits for [M&A due diligence](#), Synopsys' Black Duck Audit Services team has found that 95% of the scans they conduct reveal open source that the target didn't even know was there, and 85% of the codebases examined contain license issues.

### Quality and maintenance risks of unmanaged open source

Black Duck Audits conducted in 2018 found that 85% of the codebases scanned contained open source components that were more than four years out of date or had no development activity in the last two years. When an open source component is inactive and no one is maintaining it, no one is addressing potential issues such as weaknesses and vulnerabilities. If you use such a component, you're exposing your application to potential failure or exploit.

When abandoned open source components make their way into your applications, your developers may need to take on the responsibility for fixing any flaws or vulnerabilities that could arise in the future. Part of open source management best practices is to maintain a comprehensive, up-to-date inventory (also known as a [software bill of materials](#)) to flag components that are updated infrequently and may require hands-on attention or replacement.

Do you know whether the open source components in your applications are up to date, with all crucial patches applied? Will you know tomorrow?

It's impossible to patch software you don't know you're using. You need full, accurate, and current inventory of the open source in your applications.

## Security risk of unmanaged open source

Reports show that, on average, 67% of commercial applications contain open source security vulnerabilities.<sup>2</sup> The NVD reported [over 17,000 vulnerabilities](#) in 2019 alone; that's more than 45 vulnerabilities every day!

Because of the ubiquity of open source use, attackers see popular open source components as a target-rich environment.

Only a handful of open source vulnerabilities—such as the [Heartbleed](#) vulnerability affecting OpenSSL—are ever likely to be widely exploited. But when such an exploit occurs, the need for open source security becomes front-page news—as it did with the Equifax data security breach of 2017, in which attackers exploited a vulnerability in the open source framework Apache Struts. A contributing factor to Equifax's breach was the company's lack of a comprehensive IT asset inventory. "This made it difficult, if not impossible, for Equifax to know if vulnerabilities existed on its networks," a report on the incident concluded. "If a vulnerability cannot be found, it cannot be patched."<sup>3</sup>

## Managing your open source use: The DIY approach

If you use any open source at all, you need processes to manage your developers' use of open source components, evaluate your quality, security, and license risks, and track future developments. Below we outline six steps to create a manual system that can help you control your open source use and monitor your risk.

### Step 1: Inventory your open source

You can't manage what you're not tracking. So the first step is to create a software bill of materials (BOM), or an inventory of all open source components your teams use to develop software.

A complete and useful open source inventory must include all open source components, the versions in use, and download locations for each project in use or in development. It also needs to include all dependencies, or the libraries your code is calling to, as well the libraries those dependencies are linked to.

Consider the size of your development teams. You have a better chance to accurately track open source use if your development team is small and in one location. If you're using third-party developers, you'll need to be confident that they'll be as diligent about code inventory as your internal team is (or should be). Having a larger team or more teams can quickly make the inventory process unwieldy and more prone to errors and omissions.

Can you assure developer compliance? While your development team may agree to document their open source use as it happens, they are more likely to record it after the fact, with inaccuracies, partial listings of components, missing information on versions or download locations, unlisted components that entered via dependencies, and so on.

Can you reliably block unwanted components from your code?

## Step 2: Control your open source

Once you inventory all open source in use in your applications, you'll want to ensure you can control the use of additional open source components down the line. By having clear policies and procedures in place around the introduction and documentation of new open source components, you'll ensure you're controlling what enters the codebase and that it complies with company policies. But creating these policies, maintaining them, training developers, and staying on top of requests to approve new open source components for use is time consuming and difficult to implement.

Even if your organization feels confident that you're effectively tracking the open source you have in use, you may still be worried about how to govern its continued use and whether the written policies are enough. If you're using a manual process, you'll need to be sure your developers are following the policies you've set. You'll also need to evaluate whether those policies are efficient or are instead hampering your agile DevOps [SDLC](#).

## Step 3: Ensure license compliance

If you build packaged, embedded, or commercial SaaS software, open source license compliance is a key concern. You'll need to determine the license types and terms for the open source components you use and ensure they're compatible with the packaging and distribution of your software.

Using your BOM of open source components, you'll want to compile detailed license texts associated with those components so that you can flag any components not compatible with your software's distribution and license requirements and generate a license notices report to include with your shipped software. Unfortunately, there's no uniform approach to component license documentation. You'll need to research the licenses for each component you're using.

You may also want to involve your organization's general counsel—or seek outside legal advice—as understanding licensing terms and conditions and identifying conflicts among various licenses can be challenging for those not familiar with legal terminology. You'll want to get this right the first time, especially if you build packaged or embedded software, as license terms are often more explicit for shipped software and harder to mitigate after the fact.

## Step 4: Ensure code quality

Security and licensing concerns aside, how do you know whether you're using high-quality open source components? Are you using a current version of the software? Is it the most stable? Is the component actively maintained by a robust community?

If you're taking a DIY approach, you'll need to go to GitHub or the distribution source for each component to research project activity and potential alternatives. However, there's no universal scoring mechanism to aid in comparing different components or versions. So research can be both time consuming and subjective, a reason why many organizations struggle with effective open source governance.

Do you know whether you're using high-quality open source components?

## Step 5: Map open source to known vulnerabilities

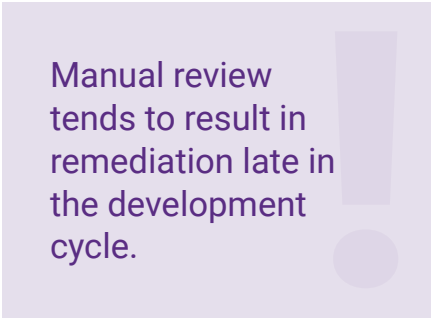
Your next step is to compile a list of known vulnerabilities that have been reported against the open source components you've inventoried.

Most DIYers will use the U.S. government vulnerability disclosure database, the [National Vulnerability Database \(NVD\)](#), as their primary source. But be aware that not all vulnerabilities are reported to the NVD. Also, the format of NVD records often makes it difficult to determine which versions of a given open source component are affected by a vulnerability.

Other useful sources of information include project distribution sites, such as those maintained by the [Debian](#) and [Python](#) projects. Security blogs and message boards, such as the [US-CERT alerts page](#) and [Google's security blog](#), should also be part of your daily vulnerability research.

"Daily" is the key word here. New vulnerabilities are uncovered literally every day. As stated previously, the NVD reported over 17,000 vulnerabilities in 2019, over 45 per day. Mapping your open source against known vulnerabilities must be a continuous process to be effective.

One challenge you'll face is the sheer volume of data that you'll need to sift through. Others include prioritizing which vulnerabilities you should address immediately, and which you can safely ignore, and then mapping vulnerabilities back to their specific locations in your code.



Manual review tends to result in remediation late in the development cycle.

## Step 6: Manage risk and monitor for new threats

Once you've identified licensing, component quality, and vulnerability risks in your open source, it's time to prioritize those risks and address them. You'll need to determine what remediation needs to be done, assign the remediation work to the appropriate people, and track the remediation process: what's being reviewed, what's been reviewed, what's been fixed, what fixes have been deferred, and what's been patched.

Challenges you can expect to face include time and cost issues. Manual review tends to result in remediation late in the development cycle, when the cost to fix is high and release deadlines need to be met. Manual review processes are also incompatible with the rapid pace and automation at the core of modern agile build and continuous integration environments.

Agile requires that you perform all the discovery processes described above continuously and integrate review and approval mechanisms into your SDLC. Without the assistance of tools to automate these processes, many organizations struggle to implement efficient mechanisms.

The job of open source management doesn't stop when the application ships. You'll need to continue to monitor for vulnerabilities as long as the application is in use. Most vulnerabilities aren't reported for months or even years after they are introduced into a component.

# SCA: A better, faster, stronger path to open source management

You probably have concerns about the amount of work open source management seems to entail. A manual approach to open source management has many drawbacks. It not only requires a significant investment of time, often for dubious results, but also affects developer productivity and leads to higher development costs.

But there's a better alternative. Smart organizations in the business of building software know they need to simplify their open source management, not further complicate their developers' lives. The key is software composition analysis to automate open source management. Managing open source with an SCA tool enables you to generate complete, accurate open source inventories, protect against open source risks, and enforce open source use policies.

Black Duck SCA is a comprehensive solution for managing license compliance, code quality, and security risks that come from the use of open source in applications and containers. It enables organizations to control open source across the software supply chain and throughout the application life cycle, set and enforce open source use and security policies, automate policy enforcement with DevOps integrations, prioritize and track open source remediation activities, and continuously monitor for new issues and alert teams if they arise.

With Black Duck SCA, it's simple to:

- Quickly build an accurate inventory of the open source your applications include.
- Deter the risk of open source license compliance.
- Safeguard your intellectual property.
- Protect code quality.
- Find, prioritize, and fix security vulnerabilities.
- Set and enforce open source use policies.
- Automate policy enforcement with IDE and DevOps integrations.
- Prioritize and track remediation activities.
- Continuously monitor for new threats.
- Configure your open source use policies based on a comprehensive array of criteria, including license type, vulnerability severity, open source component version, and more.

## Better: Get a complete view of the open source in your codebase

Black Duck SCA provides a complete, accurate view of all open source in your applications and containers by combining four methods of open source scan technology:

- Dependency analysis
- File system scanning
- Snippet matching
- [Binary analysis](#)

With Black Duck, you can build an open source inventory that is always up to date and accurate.

## Better: Eliminate open source license noncompliance

Black Duck SCA uses the industry's largest open source [KnowledgeBase™](#) to identify which licenses are relevant to the open source in your applications. The Black Duck license compliance module enables you to set policies, whitelist and blacklist, enable approval workflows, automatically generate notices reports, and automate open source management in the SDLC.

With Black Duck, you can identify all potential license obligations for the open source in use, down to the snippet and dependency level, to ensure open source license compliance for your applications.

## Faster: Continuously monitor the quality and security of the open source you use

Black Duck SCA can help your development teams avoid delays and cost overruns with risk metrics laser-focused on open source code quality and security.

Black Duck quickly identifies known security vulnerabilities, associated licenses, and code quality risks. Black Duck operational risk information uncovers a component's level of risk on the initial scan and continuously monitors the component to ensure it remains up to date and active.

- Identify component quality risks.
- Monitor for new issues in development and production.

Black Duck analyzes both source and binary code, so it can scan virtually any software, including desktop and mobile applications, embedded system firmware, and more. And with Black Duck Security Advisories, advanced proprietary research on open source vulnerabilities, you gain a complete picture of the security risk of the open source in your software.

- Map components to known vulnerabilities.
- Monitor for new vulnerabilities in development and production.
- Prioritize and track remediation activities.
- Scan virtually any software, with or without access to source code.

Integrations for each stage of the SDLC ensure that different scanning methods are run at the right time to provide maximum accuracy without slowing down development.

- Rapid Scan can be used by developers in the IDE to instantly identify policy violations before building or merging into release branches.
- CI/CD tool integrations run dependency, snippet, and codeprint analysis at build time.
- Binary analysis works to scan build artifacts and dependencies entering and leaving binary repositories.

With Black Duck SCA, you can configure your open source security and use policies based on a comprehensive array of criteria, including license type, vulnerability severity, open source component version, and more. You can also enforce development policies with automatic workflow triggers, notifications, and bidirectional Jira integration for accelerated remediation initiation and reporting.

## Stronger: Combine Black Duck SCA with Coverity SAST

Coverity SAST is a critical part of any [application testing](#) toolbox, but organizations need to further strengthen their software development strategy with a robust SCA solution.

Boost your software development process by adding in Black Duck SCA, a comprehensive solution for managing open source security, license compliance, and code quality in applications and containers.

### Resources

1. Synopsys, [2019 Open Source Security and Risk Analysis](#), 2019.
2. Ibid.
3. Permanent Subcommittee on Investigations, [How Equifax Neglected Cybersecurity and Suffered a Devastating Data Breach](#), Committee on Homeland Security and Governmental Affairs, U.S. Senate, accessed Jan. 24, 2020.



# The Synopsys difference

Synopsys helps development teams build secure, high-quality software, minimizing risks while maximizing speed and productivity. Synopsys, a recognized leader in application security, provides static analysis, software composition analysis, and dynamic analysis solutions that enable teams to quickly find and fix vulnerabilities and defects in proprietary code, open source components, and application behavior. With a combination of industry-leading tools, services, and expertise, only Synopsys helps organizations optimize security and quality in DevSecOps and throughout the software development life cycle.

For more information, go to [www.synopsys.com/software](http://www.synopsys.com/software).

**Synopsys, Inc.**

185 Berry Street, Suite 6500  
San Francisco, CA 94107 USA

**Contact us:**

U.S. Sales: 800.873.8193

International Sales: +1 415.321.5237

Email: [sig-info@synopsys.com](mailto:sig-info@synopsys.com)