Know What's in Your Code: Secure Your Software Supply Chain

Overview

Digital transformation is reshaping the way organizations operate. Whether you're one of the thousands of companies that sell software or one of the millions that use it to run your business, your ability to innovate and deliver value to your customers is powered by secure, reliable software. That's why every business is a software business.

For your business to thrive in today's challenging market, guaranteeing the quality and security of your software is crucial. But knowing what's in your code is not as simple as it sounds. Modern applications comprise a complex mix of proprietary and open source code, APIs and user interfaces, application behavior, and deployment workflows. Every point of this software supply chain contains security issues that can leave you and your customers at risk.

As the recent "Open Source Security and Risk Analysis" (OSSRA) report demonstrates, 84% of scanned codebases contained at least one open source vulnerability, and 74% had at least one high-risk vulnerability—a 54% increase over the previous year. Further, software supply chain attacks are getting more sophisticated; they now include attacks that inject malware and malicious packages into the software development life cycle (SDLC), which can transfer risk all the way down to the end user.

Identifying, tracking, and managing third-party code effectively is crucial for a successful software security program, and key to strengthening the security of the software supply chain. However, the sheer quantity of code most businesses are buying, using, and developing makes these tasks increasingly difficult.

Companies also need to take license compliance into account as part of their software supply chain security program. Open source code may not cost anything up front, but it retains license obligations that can all be difficult to identify and interpret. License compliance risk poses its own form of security vulnerability. The 2024 OSSRA report relates that over half the codebases examined contained open source with license conflicts. Such issues have been the cause of lawsuits against businesses that have used open source software without fulfilling its license requirements.

Black Duck® provides a full range of software supply chain security solutions to detect, track, and manage open source in source code, containers, and artifacts. In addition, Black Duck tools can help your organization evaluate dependencies (code your software depends on to operate) for security vulnerabilities, IP conflicts, poor health and quality, and malicious behavior.

Black Duck software composition analysis (SCA) enables you to import third-party Software Bills of Materials (SBOMs) and evaluate them for component risk, as well as build your own SBOMs automatically, in industry-standard formats, with CI/CD tool integrations and APIs. With these capabilities, teams can establish complete software supply chain visibility, identify and mitigate risk, and align with customer and industry requirements.

Here are some ways to begin ensuring that your software supply chain and codebases are secure.

Build secure code

The way to start is by building secure code, but this is trickier than it seems. In-house development can harbor vulnerabilities. The OWASP Top 10 list, celebrating its twentieth anniversary this year, details the most common vulnerabilities. For example, cross-site scripting and SQL injection vulnerabilities pose substantial risks to software integrity, and they are nearly always on the OWASP Top 10 list, but they can be too complex to spot with manual code review.

Secure coding practices are particularly pertinent for commercial application developers, given their accountability for the software they distribute to customers. Those developers must also be aware of the security vulnerabilities that come from open source and third-party code.

Some keys to building secure code include

- Giving developers tools at their desktop to help detect weak or insecure code as early as possible
- Using automation and policy to accelerate remediation and avoid pushing issues downstream
- Continuously monitoring and identifying known vulnerabilities in all dependencies, including open source and third-party components

Protect against malware

Increasingly sophisticated attackers are no longer just waiting for vulnerabilities to exploit. Instead, they are planting malware in popular repositories such as PyPI, GitHub, and npm in order to engage in intentional, targeted, and clever attacks. A recent ReversingLabs report found a 1,300% increase in threats circulating via open source package repositories between 2020 and 2023. That includes a 400% increase in threats found on the PyPI platform in 2023 alone. High-profile supply chain attacks like the recent xz Utils malware incident demonstrate the growing sophistication of these threats.

Clearly, malware attackers seek to "poison the well" from which development teams draw. And by working this way, they don't have to wait for new vulnerabilities to arise. They are infecting commonly used code sources and creating a very broad set of impacted organizations. The most frequent victims are usually the end users of the software.

The 2020 SolarWinds hack is a classic example of a malware attack. Malicious actors injected malware into an update of Orion, the SolarWinds IT performance monitoring system. When customers updated their systems, they unknowingly installed a back door through which hackers accessed users' accounts and system files. The problem spread to tens of thousands of SolarWinds customers before it was detected, and it cost SolarWinds $40 million.

Here are some ways to protect your codebase against malware.

- Secure your SDLC and development pipeline by performing security testing as your developers work, and using a tool like Black Duck Code Sight™ to deliver remediation advice right into your developers' desktop.
- Perform post-build analyses to detect the presence of malware, such as suspicious files, potentially unwanted applications, and unusual file structures.
- Continuously monitor for exposed secrets, malware, and malicious packages in the SBOMs you generate, as well as those you import.

Prepare for AI in code

Generative AI (GenAI) uses deep-learning large language models (LLMs) made from enormous volumes of pre-existing code to create new code. Developers use AI coding tools in several ways. Some use AI to autofill and complete lines of code as they're typed, while others are putting code comments as prompts and asking AI to build full methods or functions.

As GenAI excitement built, there was a rising expectation that the code it generated would be free from license and vulnerability issues and bugs. However, the reverse is true. Because the code used to train GenAI is imperfect, the code it generates is as well. As noted earlier, the OSSRA report found vulnerabilities in 84% of examined codebases and license conflicts in 53%. Further, AI tools usually lack vital context of the overall program, which can also introduce security vulnerabilities.

Potential challenges to GenAI code include

- Inadvertently incorporating license-protected code without contextual licensing information, which leads to intellectual property conflicts
- Introducing code that mirrors the security vulnerabilities typical of human developers

Source code security issues, like those in the OWASP Top 10 list, can be complex and hard to discover with manual review. This is why most development teams use static analysis testing tools, which can run through an application, gather context of information flow, and uncover problematic flaws that can lead to vulnerabilities. AI-generated source code should be checked for security vulnerabilities like any other code.

Although there's reason to be careful with GenAI code, there's no reason to let fear of these risks prevent organizations from adopting AI-generated code. Black Duck has several ways to help organizations realize the benefits of AI-generated code while managing the risks.

- Black Duck SCA snippet analysis helps teams find snippets of open source or other license-protected code that have been introduced by GenAI tools, but that are without license context or inclusion in manifest files.
- Black Duck static analysis helps teams find and fix security and quality defects in source code written by developers and AI coding assistants.
- Other Black Duck application security testing solutions help teams ensure that they are building and delivering secure software their customers and users can trust.

How Black Duck Can Help

Black Duck is the only vendor with a comprehensive portfolio that offers complete visibility into the software supply chain, giving you the ability to act on that visibility and perpetuate it with streamlined SBOM generation. This enables you to show that you are building secure applications and doing your due diligence to manage and identify software supply chain risk, so you can build trust with your customers.

Learn more about how Black Duck can help your team effectively manage supply chain security across the SDLC.

About Black Duck

Black Duck® offers the most comprehensive, powerful, and trusted portfolio of application security solutions in the industry. We have an unmatched track record of helping organizations around the world secure their software quickly, integrate security efficiently in their development environments, and safely innovate with new technologies. As the recognized leaders, experts, and innovators in software security, Black Duck has everything you need to build trust in your software. Learn more at www.blackduck.com.