REVERSINGLABS

Software Supply Chain Security and the SOC: End-to-End Security is Key

AS ATTACKERS EYE SOFTWARE SUPPLY CHAINS, DEVELOPERS NEED TO COLLABORATE WITH SECURITY OPERATIONS CENTERS (SOCS) TO HELP SPOT DEVIATIONS FROM EXPECTED BEHAVIOR IN PRODUCTION CODE

The SolarWinds supply chain attack, which distributed malicious code in SolarWinds' Orion software update to about 18,000 of its customers, demonstrated the need for organizations to have capabilities for testing software not just during the development lifecycle, but all the way through deployment and into production. "What SolarWinds exposed was that shifting left alone is not enough to reduce the attack surface," said Mat Mathews, DevOps Advocate at ReversingLabs. "It brought awareness to the need for end-to-end pipeline security."

> What SolarWinds exposed was that shifting left alone is not enough to reduce the attack surface. It brought awareness to the need for end-to-end pipeline security.

Mat Mathews
DevOps Advocate, ReversingLabs

To address the shift in threats and attacks that SolarWinds signaled, organizations need to update their security processes by shifting development security "right" to evaluate not just raw source code, but fully built, production-ready applications. At the same time SOC teams need to follow the rest of the security industry: shifting "left" and working closely with development teams to improve software security assurance, Mathews said.

For the SOC that means first understanding the baseline for normal code, file, and network behavior, monitoring for deviations from that behavior in the running code, and reporting that back to the development teams.

Why switch things up now? First, it is important to understand the history of supply chain attacks and why they are gaining prominence.

The Growing Software Supply Chain Threat

Attacks targeting the software supply chain and enterprise CI/CD pipelines have increased significantly since SolarWinds. There have been more attacks on the software supply chain since 2018 than in the previous 40 years combined, according to a ReversingLabs historical analysis of incidents.

Some of the attacks—such as one on software publisher Codecov and another on AsusTek—were similar to the SolarWinds breach in that they targeted the victim's development environment. Many others involved attempts to indirectly poison internal build environments via malicious packages published to widely used public code repositories such as npm and PyPI. In those attacks, threat actors use tactics such as typosquatting (registering common misspellings of a target organization's domain) and dependency confusion to trick developers and automated pipeline tools into downloading and using the malicious packages in their software.

For example, in April 2022 researchers from ReversingLabs discovered more than two-dozen malicious packages impersonating, or typo-squatting, widely used JavaScript modules published on npm package manager. The packages contained obfuscated jQuery scripts for stealing form data from individuals who used applications where the packages had been deployed. Some of those poisoned packages were downloaded thousands of times and installed in mobile, desktop and web applications.

In a separate February 2021 incident, independent security researcher Alex Birsan breached the internal systems of 35 major companies by taking advantage of the fact that developers at those organizations used both internally developed packages and open-source packages for building software. Birsan demonstrated how an attacker could easily compromise the build environments at those companies by planting malicious code in a public repository and giving it the same name as an internally hosted, privately developed package at the target organization. Among the firms breached in Birsan's seminal dependency confusion attack were Microsoft, Apple, Uber, and Netflix.

THE DEMOCRATIZATION OF SUPPLY CHAIN ATTACKS

Today it no longer takes nation-state smarts or resources to pull off a supply chain attack like SolarWinds, says Tomislav Peričin, Co-founder and Chief Software Architect at ReversingLabs. "You don't need to have a large budget or nation-state resources to make an impact. Anybody can sneak a software package in and target not just the developer but also the user. It is an escalation in supply chain attacks."

> You don't need to have a large budget or nation-state resource to make an impact. Anybody can sneak a software package in and target not just the developer but also the user. It is an escalation in supply chain attacks.

Tomislav Peričin
Co-founder and Chief Software Architect, Reversing Labs

The bad guys have taken notice. An analysis of data in the National Vulnerability Database (NVD), conducted by ReversingLabs, revealed that attacks on public code repositories such as PyPI and npm increased 289% since 2018. The common goal in many of these attacks is to distribute information-stealing malware, cryptominers and malicious backdoors to downstream customers. The threat posed by these attacks is even more significant when you consider the growing dependence on open-source code in enterprise software development.

According to research from the Linux Foundation, 70-90% of modern applications include code from open-source projects. Free and open-source software has become a vital component of software codebases across nearly all industries and across both the public and private sector.

MANY AVENUES FOR ATTACKS

Vulnerable and poisoned open-source components and direct attacks on development environments are part of a broader set of software supply chain security concerns. As the National Counterintelligence and Security Center (NCSC) of the U.S. Director of National Intelligence (DNI) notes, software supply chain attacks take multiple forms. They can involve anything from simple detection tactics like typo-squatting and spoofing of legitimate products to surreptitiously accessing and modifying an organization's source code using sophisticated means.

"Adversaries may seek to exploit tools, dependencies, shared libraries, and third-party code in addition to compromising the personnel and infrastructure of developers and distributors," the NCSC notes. Attacks can target software products at any stage of the software development lifecycle and be used to maintain persistence in a target environment in order to conduct surveillance or enable sabotage.

DevOps teams are aware of the risks but are inadequately prepared to address them. In a 2022 ReversingLabs survey of more than 300 technology professionals at software development companies, 98% of respondents said the use of third-party software and open-source code had heightened security risks in their organizations.

In that survey, threats hidden in open-source repositories ranked as second most critical, just behind vulnerabilities in application software and operating systems. Eighty-seven percent said software tampering could result in a security compromise in their organization. Despite that, just 37% said their organizations had the ability to detect such tampering across the software supply chain. "The software supply chain is where attacks are going to happen in the future, but many organizations don't have a strong sense of the types of code within their organization and where that (code) is inherited from," says Gregory Crabb, Founder, 10-8 Cyber and former CISO of the United States Postal Service.

> The software supply chain is where attacks are going to happen in the future, but many organizations don't have a strong sense of the types of code within their organization and where that is inherited from.

Gregory Crabb
Founder, 10-8 Cyber

A Need for Visibility

The primary focus of a shift-left approach has been to build capabilities for detecting and mitigating errors and vulnerabilities in code as it is developed and readied for release. It emphasizes the use of technologies like Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) to verify the security of code as it is developed.

Such approaches are becoming common. In a May 2022 Gitlab survey of over 5,000 software professionals, 53% of respondents said they run SAST scans to identify and address potential issues in static code that could result in a security vulnerability. Fifty-five percent said they run DAST scans against operational code to check for potential security flaws related to run-time issues such as authentication, user interfaces, and sessions. And 56% said they perform dependency scans to check for known vulnerabilities in the open-source components they use.

Shift-left testing is widely perceived as giving DevOps teams a way to maintain the cadence of application delivery that modern businesses require while reducing the costs associated with detecting and remediating security issues in code. More than half of the respondents in the GitLab survey said they shifted left to enable better application security.

Unfortunately, traditional application security testing and source code analysis capabilities don't provide deep enough visibility to detect tampering and behavioral changes of finished binaries, Peričin says. While static analysis and vulnerability scanning tools work at the source code level, they can't detect changes or anomalies in an application's behavior when the program is running.

Concerns over software supply chain attacks have also heightened enterprise interest in software composition analysis (SCA) practices. SCA platforms help organizations generate a software bill of materials (SBOM) that identifies all of the open-source components that might have been used to build a particular application and their dependencies with other components. An SBOM can help your organization identify the provenance of open-source components, license status, usage restrictions, whether they have been deprecated or contain vulnerabilities, and even if they have been tampered with.

In the U.S. government sector, SBOMs are already a requirement for vendors selling software to federal agencies, thanks to a May, 2021 Cybersecurity Executive Order issued by the Biden administration. The order requires software publishers to ensure that comprehensive machine-readable SBOMs are available for all products and classes of software that they plan to sell to the federal government.

The executive order (EO) directed the National Institute of Standards and Technology (NIST) to develop guidance for implementing these requirements. In July 2021, the US Department of Commerce, working with the National Telecommunications and Information Administration (NTIA), published a set of minimum elements (PDF document)—as required under the Biden EO—that organizations must include in an SBOM. A September, 2022 memorandum (PDF document) from the White House Office of Management and Budget set out specific deadlines for federal agencies to ensure that all software sourced from external software makers and sources have SBOMs that comply with NIST's guidance.

Unfortunately, while these measures enhance software security, they're not sufficient in themselves to prevent all supply chain attacks. Notably, tools like SAST and DAST can't detect the kind of tampering that occurred in the attack on the SolarWinds Orion code. Traditional application security testing and source code analysis can't detect software tampering in compiled code, nor can it detect compromises of development pipelines that inject malicious components into production code. One consequence of this lack of visibility: Four in 10 software packages currently in production have at least some tampering within them, according to a ReversingLabs survey on risk tampering.

IT'S NOT (JUST) ABOUT VULNERABILITY SCANNING

Security experts agree that shift-left approaches enable better software security, but software supply chain protection isn't just about detecting vulnerabilities during development. "Software supply chain security is about understanding the entire threat landscape, who is actually wanting to implant malicious code within your software and how and what they could get out of it," Peričin said.

> Software supply chain security is about understanding the entire threat landscape, who is actually wanting to implant malicious code within your software and how and what they could get out of it.

Tomislav Peričin
Co-founder and Chief Software Architect, Reversing Labs

For example, development ecosystems have become heavily dependent on open-source and third-party components, which means that organizations must consider vulnerabilities not just in their proprietary code but in open-source components and third-party code from business partners and outsourced vendors. Increasingly, software security requires the ability to monitor for risks in proprietary code as well as open-source and third-party libraries, packages, and components—both within the enterprise and across cloud, container, and infrastructure-as-code environments.

Similarly, while SBOMs can play a big role in bolstering application security, much depends on how they are constructed and used, security experts say. As ReversingLabs has observed, the process for enabling SBOMs in many organizations continues to be manual and is primarily designed for internal development teams, not for external customers and auditors.

To be useful, the information in an SBOM also must be connected with actionable information, and all applications and systems in an environment should be able to query it. Otherwise, an SBOM is nothing more than a list of components. "It doesn't tell you if any of that is fake," Peričin says.

Multiple standards are available for communicating SBOM information—such as components, copyrights, and licenses—across organizations. The most used among these are SPDX, SWID and CycloneDX.

SPDX, or Software Package Data Exchange, is a Microsoft-developed SBOM generator that the software vendor released to the open-source community in July 2022. The Linux Foundation currently hosts the project. SWID, for Software Identification (SWID) Tagging, is an SBOM format developed by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). SWID tags identify the components in a software product, the provenance of those components, the product version, and several other characteristics. CycloneDX is an open-source component analysis platform maintained by the CycloneDX Core Working Group. Like the other SBOM formats, CycloneDX gives organizations a standard way to inventory software components and the dependencies between them.

While most organizations are aware of the value that SBOMs can deliver, most haven't adopted them, the ReversingLabs survey concluded. Just 27% of respondents said their organizations generate and review SBOMs prior to releasing software. Those DevOps teams that use SBOMs focus on reviewing open-source components and internal components and focus to a lesser extent on reviewing software components from contractors, business partners and other third-parties. More than half (54%) of respondents said their companies knowingly release software with potential security vulnerabilities in them.

TOO MUCH TRUST

The NCSC identified another problem (PDF): the software integrity protocols that shift-left tools use to verify the trustworthiness of code can themselves be exploited. Cryptographically signed code, for instance, is used as an indication that code has been approved by its developer and has not been subsequently modified. Similarly, developers and users often rely on hashing algorithms to verify software integrity. The problem is that both measures can be circumvented.

Threat actors can steal the cryptographic keys used for code signing or compromise the development process before the software is signed or hashed, the NCSC notes. One example is a 2019 attack where ShadowHacker, a sophisticated advanced persistent threat actor, modified a version of an automatic software updater belonging to computer maker ASUSTek, signed the tool with a legitimate ASUS digital signature and shipped it to thousands of customers.

> In some instances, attackers have inserted malware before the software code has been compiled and signed, embedding it behind standard security signatures. In other instances, attackers have injected malicious code through genuine updates and patches for software releases and upgrades.

National Counterintelligence and Security Center

A Better Approach: Software Security Assurance

Rather than relying solely on development groups to secure organizations from supply chain threats, developers and security operations centers (SOC) need to collaborate on software security assurance.

"While developers are responsible for ensuring clean code, it takes a cross-functional team with a specific focus on security to reduce the attack surface against supply chain attacks," Mathews says.

> While developers are responsible for ensuring clean code, it takes a cross-functional team with a specific focus on security to reduce the attack surface against supply chain attacks.

Mat Mathews
DevOps Advocate, ReversingLabs

Furthermore, SOC analysts need visibility into how code behaves across the environment, including container-based software and the processes running in them. And they must have capabilities for baselining normal code, file, and network behavior so they can monitor for and detect deviations from that baseline. Those deviations might include malicious behaviors such as the use of evasion techniques, packers, and javascript obfuscators. "You have to understand how compiled software versions are supposed to behave normally to be able to detect anomalies," Peričin says. "Knowing after the fact is not good enough because the damage has already been done."

You have to understand how compiled software versions are supposed to behave normally to be able to detect anomalies. Knowing after the fact is not good enough because the damage has already been done.

Tomislav Peričin
Co-founder and Chief Software Architect, Reversing Labs

Third-party and open-source packages present a major risk to software security. To mitigate exposure, SOCs need to scan packages in code repositories and spot packers, obfuscated code and other markers of potential tampering and malicious activity in any packages the development team plans to use. In this way, the SOC will have greater visibility over all third-party and open-source components in the environment and ensure that they have been properly configured and are vulnerability-free.

A modern software security assurance strategy should bring the SOC into the loop of continuous integration/continuous deployment (CI/CD) software development and release cycles.

As developers release new versions and leverage continuous integration systems, the SOC team should be involved in the process. They should have a comprehensive report on the components and whether each is properly configured, analyze every build artifact for behavioral differences between compiled software versions, and alert the development and app sec teams of malicious, anomalous, or potentially suspicious behavior. "Not only do you need the controls of shifting left, but you also need to be able to validate the performance of the controls and the integrity of the process," Crabb from 10-8 Cyber says.

> Not only do you need the controls of shifting left, but you also need to be able to validate the performance of the controls and the integrity of the process.

Gregory Crabb
Founder, 10-8 Cyber

MANDATED REQUIREMENTS

Many of these practices are now becoming official US Government policies—including new guidance that federal agencies can implement to address growing concerns over supply chain security.

For example, the National Institute for Standards and Technology's (NIST) Version 1.1 of SP 800-218, the Secure Software Development Framework (SSDF), and the recently updated SP 800-161r1 publication on Cybersecurity Supply Chain Risk Management call for "rigorous and predictable mechanisms" for ensuring software security, such as SBOMs, binary software composition analysis, and source composition analysis, and checking for backdoors and malicious code in open-source components.

And the requirements stemming from the Biden administration's 2021 executive order call on federal agencies to integrate cybersecurity supply chain risk management with broader enterprise risk management activities. "It encourages organizations to consider the vulnerabilities not only of a finished product they are considering using, but also of its components—which may have been developed elsewhere—and the journey those components took to reach their destination," NIST noted in announcing the new publication in May.

In September 2022 the Office of the Director National Intelligence, along with CISA and NSA ODNI, released the first in a three-part series on recommended practices for software developers, publishers and agencies acquiring software from these entities. The guidelines were developed by Enduring Security Framework (ESF) and offer recommendations for all stakeholders in the software supply chain on how to meet requirements spelled out in the May 2021 Biden Executive Order.

Best Practices for Addressing Supply Chain Risks

Publications such as those from NIST provide a detailed roadmap for implementing controls in the software lifecycle that address supply chain threats. Recommended measures range from defining security requirements for secure software development, to implementing roles and responsibilities for the mission, to which toolchains to use and how to design secure software.

Here are several other best practices that every organization should consider to address software supply chain risks.

SHIFT SECURITY TESTING RIGHT

A shift-left strategy that's primarily focused on testing software during the development process is no longer enough. To protect against supply chain threats, organizations need to also implement processes for testing right of the development process all the way through to production.

The SOC should assume responsibility for testing production-ready applications for security vulnerabilities and software tampering prior to release. Its goal should be to detect and remediate run-time vulnerabilities that might have been missed during the development phase or introduced via tampering into code after development.

It's time to start thinking of SOCs as release managers for software, Crabb says.

"My recommendation to security teams is to maximize coverage in a way that's least intrusive to development," Tomislav says. If you don't have security in place, introducing a final build verification is a good first step toward maximizing coverage. If you already have security in place, consider the following steps.

MAKE YOUR SOC AN INTEGRAL PART OF THE SOFTWARE RELEASE AND SOFTWARE INTEGRITY PROCESS

Whenever and at whatever level the development team releases code into the environment, the SOC needs to be examining it for anomalies in code and/or behavior, says Mathews. "This requires comparing the latest iteration of the software with all known previous versions for any difference in behavior."

ENSURE THAT THE SOC CAN EVALUATE CODE IN A MANNER THAT'S CONSISTENT WITH THEIR TRADITIONAL MECHANISMS FOR COMMUNICATING THREATS AND RISKS TO THE ORGANIZATION

To be responsive to supply chain threats, SOC teams need to deploy code and package analysis technology that helps them identify security issues, and that does so in the the language they are familiar with and use when detecting and responding to broader threats in the environment, Mathews says.

The SOC needs to track code from the moment it is introduced into the organization to the first point of deployment and into production. "SOC teams need to be able to evaluate code in a manner that is consistent with their traditional mechanisms for communicating threats and risks to the organization," Crabb says. "That means they need to do real-time package analysis on anything that is being put into production."

USE A PLATFORM THAT WILL PLUG INTO THE DEVELOPMENT SIDE WHILE ALSO PROVIDING VISIBILITY FOR THE SOC

The goal here should be to continue to give developers tools for analyzing code as it is written and compiled across containers and CI/CD environments. SOC teams need visibility into changes happening on the software side so they can detect malicious and accidental changes to application code and behavior. They need to do this across the entire CI/CD pipeline, from the development and build phases to testing and merge, Mathews says.

Shift Right and Collaborate

The fast-evolving nature of supply chain attacks has exposed the limitations of shift-left software security assurance approaches that focus on vulnerability scanning and remediation during the software development and release process. To address the latest supply chain threats, organizations should shift security right and begin testing fully built, production-ready applications.

Developers and the SOC must work together on software security assurance. While DevOps teams focus on secure coding practices, the SOC needs to test applications after they have been developed to ensure that the code behaves as intended, and in a secure manner.

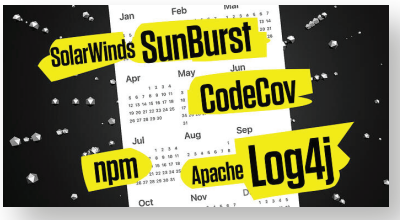
Additional Resources:

Why Malware Detection Isn't Enough Protection Against Software Supply Chain Attacks

[Download Solution Brief]

A look back at 2021: The year supply chain threats went mainstream

[Read Blog]



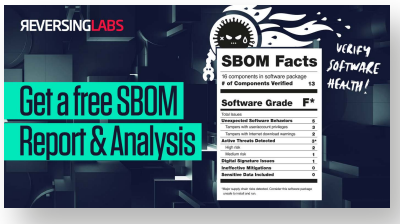
ReversingLabs supports many languages and repository packages to deliver software supply chain protection for CI/CD workflows, containers and release packages.

What You Need to Know: NIST's Secure Software Development Framework

[Watch Video]

Free SBOM: Jumpstart Your Supply Chain Security Journey

[Learn More]



Contact ReversingLabs to learn more about malware analysis solutions

REQUEST A DEMO

Worldwide Sale:
+1.617.250.7518
sales@reversinglabs.com

REVERSINGLABS