opentext™ | Cybersecurity

Position Paper

Creating a Secure Software Supply Chain You Can Trust

Be confident in everything that goes into the applications you deliver to your customers and users by evolving the security of your software supply chain. Protect the integrity of your software and SDLC with precise identification, matching, and results from proprietary research data on custom code and third-party risks.

Table of Contents

Introduction

In December 2021, researchers disclosed a critical vulnerability in Apache Log4J 2, a popular library for logging errors and run-time information. Unlike vulnerabilities in applications, however, the Log4J flaw affected software whose developers did not explicitly include the component in their application. Research by Google at the time found more than 35,800 codebases, or artifacts, on the popular Maven repository affected by the vulnerability—more than 8%, compared to a median of 0.1% for the average advisory. In addition, the average Log4J import occurred five dependencies down, meaning the component included a library that included a library that included a library that included a library that used Log4J 2.[1]

Such vulnerabilities—and the attacks that follow—are not uncommon. Approximately two years earlier, Russian nation-state actors compromised the development and deployment pipeline of network-management software provider SolarWinds, creating a backdoor in a software update that some 18,000 customers—including US government agencies—downloaded. The intelligence agency behind the hack used the foothold in those networks to compromise at least 250 networks at high-value targets.[2]

The two attacks represent two facets of the software supply-chain security of which developers need to be aware: Vulnerabilities in—or malicious attacks on—the components used to build applications, and attacks against the software and service providers that have privileged access to clients' information. Developers who use components without auditing their security run the risk of creating vulnerabilities in their own software and propagating attacks through the ecosystem.

Ensuring that software and services used by enterprises are free from known vulnerabilities requires a multi-prong strategy to lock down the myriad of processes and software components that make up your software supply chains. Companies need to trust the components and libraries used to create their own applications, track the provenance of code through the supply chain, understand their suppliers' security processes, and conduct regular behavioral analysis on running software to make sure that the code is not behaving maliciously.

Over the four years to June 2019, software-composition analysis firm Sonatype documented 216 software supply-chain attacks. The following year, the number of attacks topped 900, and the year after that, the number of attacks jumped 650% to more than 12,000.

The Problem with Supply Chains

Software supply-chain attacks have taken off in the past five years. Over the four years to June 2019, software-composition analysis firm Sonatype documented 216 software supply-chain attacks. The following year, the number of attacks topped 900, and the year after that, the number of attacks jumped 650% to more than 12,000.[3]

1. https://security.googleblog.com/2021/12/understanding-impact-of-apache-log4j.html
2. www.nytimes.com/2021/01/02/us/politics/russian-hacking-government.html
3. www.sonatype.com/resources/state-of-the-software-supply-chain-2021

In February 2021, a security researcher used typos in the names of common software components and packages to demonstrate that mistyped library names in software can lead to compromises.[4] The researcher succeeded in spreading his code to more than 35 different companies, including Microsoft and Apple, using three different languages in what he termed a "dependency confusion" attack and others referred to as "namespace confusion." By the next month, more than 10,000 copycats were found on the Node Package Manager (NPM) ecosystem, according to Sonatype, and Microsoft released a whitepaper on how to head off the attack.[5]

In another supply-chain attack, a ransomware group used three vulnerabilities in Kaseya's Virtual System Administrator (VSA) servers to compromise a number of managed service providers and then used that access to infect downstream clients with ransomware. The attack showed the capability of software supply-chain compromises to become a force multiplier: While fewer than 60 MSP customers had been exploited by the three-vulnerability exploit chain, up to 1,500 companies who subscribed to those MSPs services were compromised.[6]

Two Prongs of Software Supply-Chain Attacks

A software supply chain can be long and complex. The chain extends from applications down to open-source—or community-provided—components. The components can then include additional layers of open-source components.

A software supply chain can be long and complex. The chain extends from applications down to open-source—or community-provided—components. The components can then include additional layers of open-source components.

Overall, attacks on the software supply-chain fall into three broad categories: Compromising the development pipeline, exploiting the software operations pipeline, and vulnerabilities in a subcomponents or dependencies. An example of an attack against the development pipeline includes dependency-confusion, where the attacker targets the software used by developers, while development pipeline attacks include Keseya and Codecov. Finally, vulnerabilities in dependencies and sub-components include Log4J and Spring4Shell.

And no matter where the vulnerabilities enter the software supply chain, maintainers and companies are slow to realize the security issues. Malicious packages introduced to the software supply chain were available for an average of 209 days before someone publicly reported the vulnerability.[7] Meanwhile, software developers tended to move slowly as well, taking 205 days to deploy mitigations for vulnerabilities in their own software in 2021, up from 197 days in 2020.[8] (Lets talk about our numbers).

The result is that software supply-chain vulnerabilities are hard to detect for developers of the packages and slow to be deployed by maintainers and downstream users, making them an ideal vector for attackers and a difficult area of security for defenders.

4. **https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610**
5. **https://azure.microsoft.com/en-us/resources/3-ways-to-mitigate-risk-using-private-package-feeds/**
6. **www.darkreading.com/vulnerabilities---threats/attacks-on-kaseya-servers-led-to-ransomware-in-less-than-2-hours/d/d-id/1341496**
7. **https://arxiv.org/pdf/2005.09535.pdf** (pg. 11)
8. **www.zdnet.com/article/average-time-to-fix-critical-cybersecurity-vulnerabilities-is-205-days-report/**

A Backstabber's Knife Collection
A seminal paper, *Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks*, written by a team of European researchers collected 174 packages used in supply-chain attacks against open-source software between November 2015 and November 2019. The researchers found that each software ecosystem—the Python Package Index (PyPI), the Node Package Manager (NPM), and RubyGems—all had malicious code inserted into the supply chain. However, malicious modifications to PyPI components tended to be slower to be detected, while RubyGems tended to be quicker. In addition, malicious RubyGems tended to execute their payload during the runtime phase, while PyPI modifications tended to execute their malicious code during the installation phase. NPM packages split fairly evenly between the two tactics.

A Multi-Prong Approach to Cyber Resilience Is Necessary

Most companies have targeted different pieces of the software supply chain, focusing on software security, vendor security, or monitoring their applications for malicious activity.

The need for securing the software supply chain is no longer an option. In May 2021, the Biden administration announced Executive Order on Improving the Nation's Cybersecurity (14028),[9] which calls for removing barriers to information sharing (Section 2) and enhancing ways to improve supply-chain security (Section 4). Among the improvements, companies must work with the National Institute of Standards and Technology (NIST) and other government agencies to create a plan. Already, NIST has released a draft document[10] that calls for Software Bill of Materials (SBOMs) and enhanced security assessments of vendors. The effort will likely spread beyond the US, as global companies will have to meet the requirements to do business with the government agencies.

Securing Your Supply Chain Requires a Multi-Prong Approach
Just as the software supply chain is an interrelated combination of complex systems, the effort to secure the entire ecosystem requires a multi-disciplinary approach. Companies should bring together teams from software development, operations, legal, human resources, and business to create a holistic approach to producing resilient software.

1. **Not just technology, but people and process**
   A single—or even a group of technologies—will not solve any company's software supply chain problems. An employee culture of embracing cybersecurity along with documented—and constantly updated—processes are both required for the foundation of a secure software supply chain.

Just as the software supply chain is an interrelated combination of complex systems, the effort to secure the entire ecosystem requires a multi-disciplinary approach. Companies should bring together teams from software development, operations, legal, human resources, and business to create a holistic approach to producing resilient software.

9. **www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/**
10. **www.nist.gov/itl/executive-order-14028-improving-nations-cybersecurity/software-security-supply-chains-software-1**

2. **Protect the software development pipeline**
   While a secure development lifecycle (SDLC) is a start, companies should go beyond a pipeline designed to catch inadvertent vulnerabilities toward a resilient approach that is designed to catch changes by untrusted actors and code that behaves in anomalous ways. NIST published a document to guide developers in auditing their software—the Recommended Minimum Standards for Vendor or Developer Verification (Testing) of Software—which aims to help firms secure software used by US federal agencies.

3. **Produce high-quality software**
   Companies need to make sure that vulnerabilities are quickly detected before software is deployed and that patches applied in a timely manner. Static scanners, dynamic tests on staging servers, and interactive application security testing can all help catch vulnerabilities before they debut in publicly released software. Currently, only about half of development teams use software composition analysis (SCA) tools and static application security testing tools (SAST), while fewer—about four in ten—firms use infrastructure-as-code (IaC) or web application scanners.[11]

4. **Respond quickly to vulnerabilities**
   Even with overlapping checks and balances, vulnerabilities will get deployed to products and services. In those cases, companies should have a process in place to quickly identify, confirm, and remediate vulnerabilities.

To support a security of the software supply chain, the technology components need to work together. Cybersecurity has a portfolio of products that are tightly integrated and address the different facets of supply-chain security.

Fitting the Parts Together

To support a security of the software supply chain, the technology components need to work together. OpenText™ Cybersecurity has a portfolio of products that are tightly integrated and address the different facets of supply-chain security:

- Fortify by OpenText™—A set of tools for securing custom, open source and third party software using SAST and DAST for secret scanning to detect hard coded supply chain credentials, protect the pipeline with CI/CD as code, identifying 3rd party client side JS in web apps and more
- Debricked—This machine learning-powered tool helps development teams use open source in a smart, efficient and secure way. Through automation and clever, customizable policy setups Debricked helps your organization take a proactive, rather than reactive, approach to open source vulnerabilities, license compliance and project health.
- NetIQ by OpenText™—The NetIQ Risk Service by OpenText protects against high-risk authentication and application access requests by initiating strong or multi-factor authentication when risk scores indicate that a higher level of identity verification is needed. This can be an added layer of defense.
- Voltage by OpenText™—Discovers, analyzes, and secures sensitive data across hybrid multi-cloud IT, protects data privacy, and helps businesses satisfy compliance requirements.

11. Snyk State of Open Source Security Report, p. 21.

- ArcSight by OpenText™—AI powered Security Information and Event Management (SIEM) platform that iaccelerates effective detection and response to known and unknown threats.
- Security analytics, AI & ML—Automation is critical to be able to keep up with attackers, so Cybersecurity has incorporated machine automation and learning technologies into its products. ArcSight Intelligence by OpenText™, for example, identifies and prioritizes anomalies through user entity and behavior analytics UEBA) that may uncover threats, such as data exfiltration and attacks on DevOps accounts.

Defending the Software Supply Chain

Current software development could not exist with the open-source ecosystem. Almost all software (98%) tested by security firms makes use of open-source components, with an average of three-quarters of every codebase consisting of open-source code. This widely available ecosystem has also attracted software developers from all parts of the world and from non-technical careers.

The result is an ecosystem that is a necessary foundation for competitive software development, but at the same time, is prone to defects, errors and vulnerabilities. For that reason, companies need to not only worry about their own software development pipeline but external sources of code as well.

Hardening the Fault Points in the Supply Chain

A critical step for organizations to avoid incorporating flawed or compromised code into their own products is the ability to spot anomalies in open-source components and projects. In many cases, open-source maintainers do not have the time nor funding to enforce a secure development lifecycle on their project. Spotting warning signs—such a long time to remediate issues—can help companies to decide whether using code from such a project is worth the risk.

Spotting warning signs—such a long time to remediate issues—can help companies to decide whether using code from such a project is worth the risk.

Watch Out for the Easiest Attacks

At present, the most common attempted attack is dependency confusion. The ease with which attackers can create malicious project to take advantage of typing errors has lead to thousands—if not tens of thousands—of software project specifically created to fool developers. While the attack has a very low success rate, like phishing and spam, it only takes one user at a company to give attackers a beachhead from which to compromise other systems. In addition, online tutorials can be used as a way to propagate the names of malicious dependencies, taking advantage of novice developers.

Focus On—Detecting Anomalies in the Software Ecosystem

Debricked tracks information about open-source projects to score the projects on their security as well as give developers a measure of the project's popularity. Other metrics that can be used as a guide to the stability of a particular project include a score for the contributors.

Producing Secure Software

Knowing the current types of attacks targeting the software supply chain can inform security improvements for the development and deployment of software, but security needs to be well integrated into the process to avoid creating bottlenecks in the creation of applications and new features. Some simple process changes can have outsized security impacts while maintaining developer momentum.

Knowing the current types of attacks targeting the software supply chain can inform security improvements for the development and deployment of software, but security needs to be well integrated into the process to avoid creating bottlenecks in the creation of applications and new features.

Provide Developers the Information They Need

The shift-left movement has placed more responsibility for software security on developers, but the focus on security should be implemented in a way as to minimize any disruption to development. Security information and guidance, for example, should be presented as early as possible to both alert the developer to poor security patterns and defects that create vulnerabilities, but also as a way to train developers to recognize such coding patterns in the future.

This extends to the supply chain as well. Security metrics should be presented whenever developers import libraries or include components, so they have information about the security and stability of open-source components and don't import vulnerable components into their software, essentially propagating exploitable code. In addition, the company's overall policy on component characteristics should be available through a dashboard to provide opinionated conclusions on the projects behind the code.

Remediation: Helping Enterprises Get the Control Back

Open-source components being shared among different applications need to be remediated as soon as possible. There are different strategies followed by enterprises to remediate and cover the exposure of these identified vulnerabilities.

1. **Upgrade Component(s):** In most cases, reducing the risk will involve deploying an upgrade recommended by the tool itself. However, Upgradation can be tricky sometimes since it requires the business-critical system to be halted for a given time window. So think of removing the Vulnerable component that is not actually being used
2. **Patch Component(s):** In some cases, Developers will go ahead and fix the vulnerability by forking the Open-Source component master branch and creating a new release path for the revised and secured component which can be moved to white-list and approved software, the best practice is to perform SAST on the code now owned by the development team.

3. **Disable the vulnerable process or function:** this might be the easiest way to reduce the risk but if the vulnerable component does not impact the business functionality, it can also be disabled.

Block Known Malicious Attacks, White List Approved Software
The shift-left movement has placed more responsibility for software security on developers, but the focus on security should be implemented in a way as to minimize any disruption to development. Security information and guidance, for example, should be presented as early as possible to both alert the developer to poor security patterns and defects that create vulnerabilities, but also as a way to train developers to recognize such coding patterns in the future.

Conclusion

Software supply chains form a complex ecosystem that allow companies to innovate quickly by using common components. The strength of the ecosystem, however, can also be a weakness if attackers are able to compromise the open-source projects, the resulting components, or the vendors on which companies rely for their software and services.

Software development teams should use information about open-source projects to determine how well they follow software security best practices, monitor the projects for anomalies that could indicate a compromise, and analyze any imported code for vulnerabilities and potentially malicious changes.

Next Steps
Companies should:

- form a process to generate whitelist request(s) by the application teams and approval for the new component(s) by the security teams.
- form an interdisciplinary team to set policy for using external software in a way that allows innovation but reduces security risk,
- educate themselves on software supply chain risk,
- identify all open-source components used in current software projects to create software bills of materials (SBOMs),
- determine whether those software projects and components meet the current policy using metrics, and
- monitor for software changes that could indicate the source has been compromised.
- security advocate awareness towards the approved or whitelist components

Learn more at
www.microfocus.com/en-us/cyberres/use-cases/securing-the-software-supply-chain

Turning Data into Security Information
It is not enough to track the provenance of a particular software component and the trustworthiness of the contributors behind the software. Companies need a way to detect changes in the code of which the project maintainers may not be aware and changes in runtime behavior that may indicate maliciousness. Fortify gives companies the capability to analyze static code for defects and vulnerabilities, as well as test runtime behavior dynamically.

Connect with Us

www.opentext.com

opentext™ | Cybersecurity