WHITEPAPER

SCA Runtime Protection

Reducing Software Supply Chain Risk from Known Exploits to Future Zero-days

MERGEBASE

Table of Contents

Introduction to Software Supply Chain Security

Software supply chain security is the practice of identifying and addressing risks that are incurred when using third-party software as part of the application development and deployment process. Software supply chain threats can be more damaging than a single directed cyber-attack because they can impact thousands of different targets that are downstream in the software supply chain. Most importantly, supply chain attacks take less time and resources but scale massively, as the attacker can just be following the scripted attack vector in the published vulnerability.

One of the most recent examples of software code with a critical vulnerability is the Log4j Java library used for logging. Within 24 hours of it being reported publicly, over 200k attacks were launched! Due to the library's extensive use worldwide, this vulnerability had vast implications for millions of systems across multiple OS platforms. However, it turns out that the vulnerable "lookup" method in the Log4j library was rarely used, yet the only solution was to update the whole library.

Software composition analysis (SCA) solutions identify and provide remediation guidance for known vulnerabilities in third-party components, especially open source software (OSS) libraries. Given the popularity of utilizing OSS in modern application development, SCA is an important part of application security testing (AST) today.

While it is common to find that OSS comprises nearly 80% of code in an average enterprise application,[i] until recently, most testing focused on the 20% of the application that your team or vendor coded directly, performing only static application security testing (SAST) on their own code.

ONLY 20% OF YOUR CODE IS TESTED AND LEAVES 80% OF THIRD-PARTY CODE UNPROTECTED

This means only 20% of your code is tested and leaves 80% of third-party code unprotected, blindly assuming the components that came down from the software supply chain were secure.

Another key reason why software supply chain security is often ignored is that it can be a very hard problem to solve. The traditional method of supply chain risk mitigation has always been to build a strong perimeter around your enterprise and its line of business applications. However, now that your most important applications are Internet-facing, perimeter security is greatly diminished, especially in development and non-production environments, and is no longer able to protect you from supply chain attacks.

The common vulnerabilities and exposures (CVEs) in your software components are often publicly exposed and low-hanging fruit unless you can quickly update and patch those libraries before an attack is launched. Software supply chain attacks in non-production environments are the perfect entry point for criminals to traverse the organization, elevate privileges, and then compromise the "crown jewels"—the most important assets of your organization.

The good news is that a patch or update is generally available when a CVE is reported publicly. The bad news is that every hacker in the world can read in minutes detailed instructions on how to exploit the software on day 1 of the CVE report. Within the first week of the Log4j CVE announcement in December 2021, over 1.2M attacks were launched against the new Java vulnerability, including from nation-state actors like China. Patching can easily take weeks or months, even if you have the resources readily available, or it may never happen if your vendor is not motivated. That's right; sometimes, a patch will never be made available, putting you at lasting risk until the whole application is replaced.

So, what do you do if you are, say, a major bank using an enterprise application in production that may take months to patch? You can't close the bank on Monday or turn off your website for a few months, as you must remain open for business or lose all of your customers. If the application is internally developed, you can pressure your dev team to try to accelerate the update process. But if it is a commercial application that you purchased, you are at the mercy of that vendor to make the update. Even if the vendor provides a patch, your team would still need to deploy it into production, which takes time for testing to ensure the patch doesn't break anything else.

When your business cannot be stopped for a supply chain security vulnerability, there are a few mitigations that can be implemented, like a web application firewall (WAF) or runtime application security protection (RASP). Unfortunately, neither of these is easy, nor can they guarantee that an attacker will not subvert them to exploit the CVE in the heart of your enterprise application.

Enter SCA Runtime Protection™

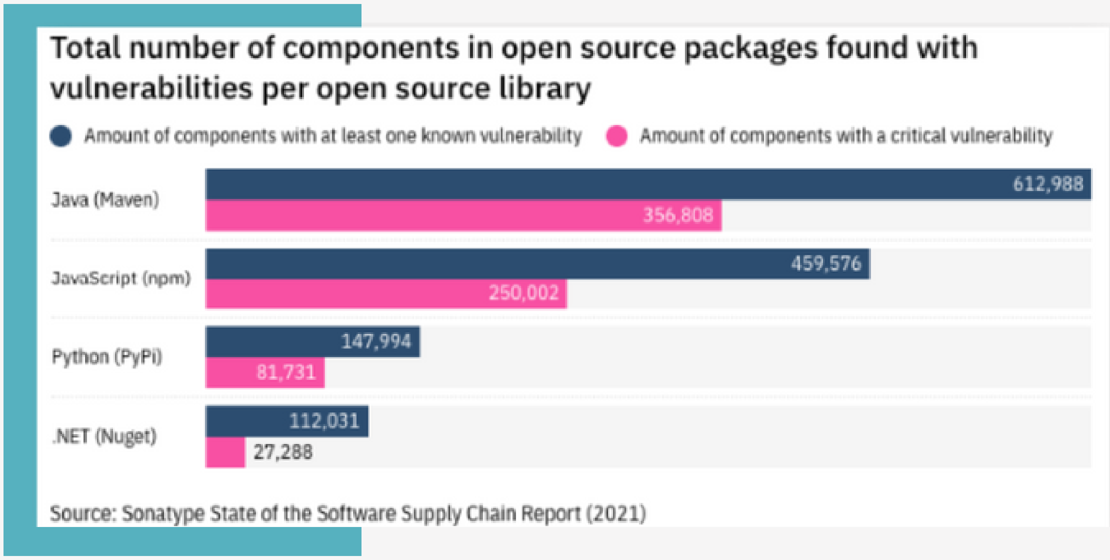
SCA Runtime Protection, a patented software supply chain security solution provided by MergeBase, protects your applications from the exploitation of vulnerable Java libraries or methods (e.g., Log4j, Struts, Jackson, etc.), regardless of patching status. This new option for blocking software supply chain attacks buys your organization as much time as needed to calmly update or patch vulnerable software libraries without rushing the development, testing, or deployment processes. In the event that a patch or update will never be available, Runtime Protection can be used indefinitely on legacy systems. What's more, its performance toll is less than 1%, so it will not noticeably slow down production systems.

Background on Open Source Software and Third-party Supply Chain Risk

To understand better why software supply chain risk is an increasingly hard problem to solve, you first need to be aware of the rapid growth of the open source software ecosystem. Only after grokking the immensity of the OSS universe and the complexity of the resulting third-party risk, will you begin to appreciate the true value of SCA Runtime Protection.

Over the last decade, we have seen exponential growth of OSS projects, resulting in a huge increase in the potential attack surface for supply chain attacks. For example, the number of public repositories hosted by GitHub exploded from 46,000 in 2009 to over 28 million by 2020.

With a three order of magnitude increase in OSS, it is no surprise that supply chain security is an increasingly important issue that the US and Canadian governments are starting to regulate. In fact, in 2021, software supply chain attacks grew by 650%, according to research by Sonatype, which recorded 12,000 incidents in 2021.[ii] Sonatype's report assessed the number of vulnerabilities across the most common open source packages and found that Java had the most components with vulnerabilities, including more than 350,000 that are deemed "critical" and could be exploited to gain root-level access.



Source: Sonatype State of the Software Supply Chain Report (2021)

In a recent global study of 1000 CIOs sponsored by Venafi, 82% say their organizations are vulnerable to attacks targeting software supply chains.[iii] Finally, in early 2022 Gartner predicted that software supply chain attacks will continue increasing over the next few years, expecting 300% more supply chain attacks by 2025.[iv]

MITRE: Common Vulnerabilities and Exposures

Common Vulnerabilities and Exposures (CVE) is a global catalog of known security threats. CVE is sponsored by US-CERT, within the Department of Homeland Security Office of Cybersecurity and Information Assurance (OCSIA). MITRE founded the CVE system over 20 years ago and maintained the CVE dictionary and public website, functioning as Editor and the Primary CVE Numbering Authority (CNA). There are over 250 CNAs globally, which include nearly every major software company in the world. In 2022, there have been 24,509 CVEs reported so far, and there are over 200,000 actives in the National Vulnerability Database.

Patching Delays Result in Extended Vulnerability Windows

Generally, just before a CVE is published globally, the software creator releases a patch or update to fix the security bug in the software library or component. Most of the time, the patch is just updating a single function or method within the library, so it would seem quite simple to fix the problem by running the update and applying the patch. However, nothing is simple in the complex world of interconnected software, networks, and systems we have today.

One minor change often can have unexpected results in complex systems. So, before a patch is applied in production, it must first be applied in a test or staging environment to see what this "fix" might break if anything. And when you have hundreds or thousands of systems needing the patch, each environment might be slightly different. Thus, the testing can be very complex and time consuming. Therefore, a patch that might only take seconds to apply could easily take months to implement across all production environments. In the hackers' world of computer security, months are like years, giving attackers plenty of time to deploy systems to try to exploit the new CVE across thousands, if not millions, of targets.

Zero-day or 0day: Either Way It's Spelled, It's Bad

A zero-day (0day) exploit is an attack targeting a software vulnerability that is unknown to the software vendor or the users of the software. It is very hard for developers and security experts to find all security flaws so attackers expect that they exist and expend substantial effort to discover them. The result is an "arms race" between the attackers and the security community, where zero-day flaws that offer remote code exploits can be sold on the dark web for $100,000 or even millions.

Zero-day exploits provide a huge benefit to attackers because security defenses are built around known exploits, so targeted attacks based on an as-yet-unknown flaw can go unnoticed for months or years. The success of a zero-day exploit attack depends on the vulnerability window—the time between an exploit's discovery and its patch. Even a known vulnerability can have a lengthy vulnerability window if its patch is difficult to develop and/or deploy.

Java: Largest Attack Surface & Attackers' #1 Favorite Target

Java is the #1 enterprise programming language in the world today. With more than 10 million developers using it, Java is installed on 56 billion Java Virtual machines (JVM) globally and 34 billion cloud-based JVMs.[v] As Java is ubiquitous and the dominant language used for mission critical systems in enterprises, it is no surprise that Java-related vulnerabilities are the most numerous and most lucrative for attackers.

Java exploits are popular because of Java's platform independence and ability to run on every operating system, offering the widest attack surface of any platform. In fact, as we saw in the Sonatype research chart above, Java has more critical vulnerabilities than JavaScript, Python, and .NET combined.

Java is deployed widely in both web browser plug-ins and Java-based enterprise applications. In addition, most users do not regularly update the Java Runtime Environment (JRE), and patches released for known Java vulnerabilities are rarely applied immediately. That means most devices are running an outdated version of Java with a known flaw for some time, which puts them at a higher risk of exploitation.

As a result of the massive attack surface and slow patching hygiene, Java is the number one target for hackers looking to get a foothold into your applications to start their attack toward your valuable data. While it is always good to have a defense-in-depth security strategy, if you can protect your public-facing Java applications from exploitation, chances are the attackers will move on to their next target.

Break Your Attackers' Kill Chain with Runtime Protection

Once a CVE is published, the clock starts ticking, and the arms race is on. Attackers are generally the first ones to read and understand the implications of a new CVE that is rated "high" or "critical" severity. Then they work quickly to add this shiny new software supply chain exploits to their arsenal and start making their attack plan.

Product security teams who are practicing good application security testing as part of their DevSecOps processes should be alerted by their software composition analysis platform if they are affected by the new CVE. They can begin preparing to update or patch the hackable software library and make a new build of their application that will not be vulnerable to the CVE. Hopefully, this can be done in weeks, not months, although it will probably take a month or more to roll out the new version.

If you are a customer of the application, you need to wait until your vendor releases the update before you can apply the patch. You may not even know that you are at risk until you get the update (a month or more later) unless you have an SBOM from that vendor and were able to match the CVE to a component listed in your application's SBOM.

While vigilant vendors immediately release updates, others take a long time, and some never acknowledge the existence of the vulnerability. Meanwhile, attackers are rolling out the siege engines to start executing hundreds of attacks per day. It may not be long before your supply chain component is the next target in the attacker's campaign.

Whether you develop or buy the application, patching takes more time than exploiting the CVE at scale. What's needed is a way to buy more time so that patching isn't rushed and prone to breaking line-of-business production systems in the sprint to stay secure.

Runtime Protection Protects Your Business Now, Buying Time to Patch

MergeBase is the only SCA platform that can also block Java exploits without patching, in addition to its highly accurate software composition analysis and complete SBOM and container support. With MergeBase's patented SCA Runtime Protection capability, you can choose to monitor or block any Java-related CVE, either the whole software library or the specific vulnerable function.

With Runtime Monitoring, you can determine if your application is utilizing the method impacted by the CVE. If so, you can set an alert to be notified, so if someone suspiciously utilizes it, you can track them. If you are not using the vulnerable Java method, you can block it from ever executing, completely eliminating your risk of that known exploit.

In the past, the only option was to wait for a patch to be applied and hope it did not break your application when updated. After learning that a Java CVE like Struts or Log4j impacted your software supply chain, you had to rush to remediate with a new build or updated deployment via a midnight patching run as you were not safe until updated.

With Java Runtime Protection, you can now break your attackers' kill chain, buying your team as much time as needed to calmly remediate with minimal disruption to production. No more sleepless nights or frantic late-night patching runs over the weekend since MergeBase empowers you to choose to monitor or block the vulnerable Java component or function immediately.

Reduce Attack Surface from Zero-days with Runtime Monitoring and Protection

Harden your line-of-business applications and reduce your attack surface with SCA Runtime Protection. Now you can dramatically reduce vulnerability to zero-day attacks and CVEs by shutting down access to all unused third-party Java libraries and functions.

MergeBase SCA Runtime Protection empowers you to perform real-time software runtime monitoring to learn what third-party dependencies are used by your enterprise applications. This allows you to intelligently shut off the execution capability of all unneeded Java components and methods, preventing known exploits or CVEs as well as unknown or zero-day attacks.

With Runtime Protection, you can break your attackers' kill chain before they strike or even before the CVE is published. Runtime Protection is a sure way to stop attacks on Java-related CVEs and reduce your operational risk by eliminating execution access to unused or exploitable Java software libraries and functions.

Shifting Right: Putting the Ops Back into Your DevSecOps with Production Visibility

All other SCA solutions cover only the development stage, stopping at "DevSec", while MergeBase SCA gives you real-time visibility and insights into your live deployments in production, providing a full DevSecOps view from development into production.

When a new CVE comes out, if you are developing your own applications and using SCA, you can check if you are impacted by looking at your development to see which third-party software components are used in building your application. However, if you bought the application from a vendor, you probably don't know what software libraries are in the application unless you required an SBOM from them before purchasing it.

While many companies don't do their own internal software development, all businesses are responsible for the security of their customers' personally identifiable information (PII) and other business-critical information. Recent legislative changes in many countries, including the US and Canada, have put the liability squarely on the company using the application, making them fully responsible for managing their software supply chain risk. So, if your purchased application is exploitable due to a known CVE, you are responsible for the risk that that software may expose the data in your possession. Therefore, you need to know what vulnerabilities are in your software supply chain regardless of whether you are the developer or just a user of the application.

Since most software composition analysis solutions don't analyze applications that you purchase and deploy into production, it is difficult to know what you have to protect against. This is why the US and Canadian governments have recently passed laws requiring a software bill of materials (SBOM) for all applications sold to the respective governments. They want to be aware of the risks associated with the software components in the purchased applications. Many large enterprises are now also requiring SBOMs before purchase and with each updated version for commercial enterprise applications (including SaaS apps).

MergeBase SCA Platform offers complete SBOM functionality featuring both import and export and supporting both major formats (CycloneDX and SPDX). With its Runtime Monitoring, you can always know what third-party software components are in your production applications, regardless of whether you developed or purchased them.
MergeBase gives your business real-time observability into your software supply chain risk running in production. Not only will you know if a library or method is currently vulnerable to a new CVE, but you can inspect the integrity of software components to verify that they were not poisoned and injected with malicious code somewhere in the supply chain.

Close your Vulnerability Window Today

With MergeBase Runtime Protection installed vulnerabilities can be eliminated in minutes. This enables you to close your vulnerability window within a day or two after a CVE announcement is made—certainly faster than the bad actors can organize and deploy a new attack plan.

Installing Runtime Protection is Easy

MergeBase has plug-ins for all major continuous integration/continuous delivery (CI/CD) tools, and those can be installed in minutes in our pipeline. Once that is done, all new builds can have Runtime Protection in them.

Security Superglue: Low-cost, High Performance, Significant ROI

Runtime Protection is not only easy to implement, but it is quite cost-effective, both in labor and price. While patching is absolutely the right thing to do, it takes time and should not be rushed, whereas Java Runtime Protection is a nearly instant fix that can last years if needed. Between buying time to patch on the regular schedule and dramatically reducing risk of a breach before patching, no other solution for stopping software supply chain attacks offers such an attractive return on investment.

Furthermore, the performance cost to your application is less than 1% of CPU usage for blocking a software component or method, and only about 5% for monitoring. You would need to use monitoring when you have to leave that component or method enabled for production requirements, but you could set alerts to go off when it is used to verify that the user is not an attacker.

WAF vs Runtime Protection

A web application firewall (WAF) looks at incoming web traffic to identify known attack payloads and patterns of abnormal usage. When a suspicious payload or abnormal pattern is detected, this can be reported and blocked. WAF blocks IP addresses and provides customization of a set of rules in addition to real-time alerts and reporting. WAF tries to separate known bad traffic from good traffic and works to ensure information is not getting processed which is not relevant or malicious. Generally, this filtering results in a bandwidth or network performance hit that can slow down production.

While a WAF can be programmed to stop a software supply chain attack in some cases, it is not a guaranteed solution, nor is it a simple fix. A WAF blocks threats by detecting risk, but there is also the potential for false positive detections, which may cause problems that could block valid traffic. When this happens, your business is effectively taken offline, and production is halted. It is a fuzzy logic system that can lead to trouble due to unpredictable outcomes in a production environment.

Runtime Protection is very simple and precise. Either it allows a software component or specific method to execute, or it blocks that execution. It acts like your electrical breaker box. If the power is cut there, you can still flip the light switch, but without power, there will be no light. There are no false positives or slowing down production with Runtime Protection.

RASP vs Runtime Protection

Runtime Application Self-Protection (RASP) is a security technology that installs an agent into an application runtime environment. RASP then instruments and observes an application as it executes, attempting to detect and prevent real-time attacks. It benefits developers and security staff because RASP can highlight how the application is being attacked in production. Taking that feedback, developers can modify the code base to eliminate security vulnerabilities.

Unlike a WAF, which is external to the application, RASP works internally via an API or linked library and runs as an agent within the application. RASP is generally trying to detect unknown vulnerabilities that are in the process of being exploited, sounding the alarm, and, if appropriate, dynamically stopping the attack.

Like Runtime Protection, RASP is implemented for each individual application, as opposed to a WAF, which generally protects multiple applications. However, like a WAF, RASP is a fuzzy logic process that can produce false positives that could take down production. It is also doing complex processing inside the application and, as a result, can slow down the application directly, whereas a WAF is only slowing down the network traffic before it reaches the application.

Whether Reactive or Proactive, Runtime Protection is Exact and Fast

Runtime Protection is an exact binary process, either allowing execution of the third-party software component or not. There is less than a 1% performance hit and no risk of false positives or taking down production. Furthermore, Runtime Protection serves multiple purposes. It is first used to reactively shut down access to known vulnerabilities that cannot be patched immediately. It also can be used proactively to close down access to all unused software components to reduce attack surface and prevent attacks exploiting future zero-days or unknown vulnerabilities.

Omnipresent Supply Chain Risk

Everywhere you look, there is third-party risk in the software supply chain. Just as every cake is baked with supply chain ingredients like flour, sugar, and eggs, nearly every application is built with third-party components. Risk can be minimized with solid DevSecOps processes like secure development lifecycle (SDLC) management, transparent software bill of materials, and choosing vendors that transparently embrace these best practices for software security and communication openly with their customers.

When vulnerabilities are found, there are always unexpected risks and new challenges that prevent you from closing your vulnerability window as fast as you planned. Time is not your friend in these stressful situations, as risk compounds and grows over time. SCA Runtime Protection can buy you time and reduce risk by closing your vulnerability window until proper remediation can be completed, and if it can't completely close that window, you can at least use Runtime Monitoring to keep a close eye on the vulnerabilities that are still open for exploitation.

Runtime Protection: Incredible ROI and Cheap Cyber Insurance

When Equifax was hacked in 2007 due to the Apache Struts CVE, the company lost $5 billion of its market capitalization in one day, which it never fully recovered. They tried hard to patch all their systems but couldn't get it done before the attack came. They needed more time, but unfortunately, Runtime Protection was not available back then to allow them to buy time and close their vulnerability window soon enough to break their attacker's kill chain.

Developers' time is a precious resource and disrupting their production schedules to rush a new update for every new security vulnerability is not a good model. It creates alert fatigue, is costly, and delays production. Today, every organization can invest in Runtime Protection to buy time and reduce risk while reacting productively to the never-ending flood of CVEs. If the risk to production is neutralized, developers and admins don't need to pull an all-nighter to push an update or roll out patches over the weekend.

SCA Runtime Protection is an affordable solution that could literally save your organization face with its customers and billions of dollars for its shareholders. It's hard to put a price on customer trust, but without a doubt, breaking your attackers' kill chain before they arrive is the best investment and cyber insurance you can buy.

Next Steps:

Review the Customer Case Study

If you want to learn how Runtime Protection helped one of the largest financial institutions in North America to balance their software supply chain risk with production "Ops" requirements, you can download a brief customer case study here.

See Runtime Production in Action

If you'd like to see Runtime Protection in action and learn how easy it is to set up and manage, book a demonstration with MergeBase at your convenience by clicking here.

For more information:

mergebase.com

info@mergebase.com

i. The 2022 “Open Source Security and Risk Analysis” (OSSRA) report - https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html

ii. 2021 State of the Software Supply Chain: Open Source Security and Dependency Management Take Center Stage - https://blog.sonatype.com/2021-state-of-the-software-supply-chain

iii. Study: 82% of CIOs Say Their Software Supply Chains are Vulnerable - https://venafi.com/blog/study-82-cios-say-their-software-supply-chains-are-vulnerable/

iv. 7 Top Trends in Cybersecurity - https://www.gartner.com/en/articles/7-top-trends-in-cybersecurity-for-2022

v. Oracle announces Java 18, brings new capabilities - https://timesofindia.indiatimes.com/gadgets-news/oracle-announces-java-18-brings-new-capabilities/articleshow/90405259.cms