sonatype

The Evolution of Software Supply Chain Attacks

Introduction

Most businesses today depend on open source components to simplify development, lower costs, and improve code quality. According to one recent study, 77% of respondents admit to increasing their use of **open source software** over the last 12 months, with 36.5% increasing it significantly—a trend that is bound to accelerate in the years ahead.

But despite the numerous benefits that come with using open source software components, risks abound. Threat actors are increasingly launching sophisticated attacks against software supply chains and targeting open source vulnerabilities to rake in profits, knock businesses offline, and steal private data.

Most businesses are still in the dark about this growing problem. As more companies turn to open source solutions, the majority remain unaware of security issues lurking in their software supply chains. As it turns out, 1.2 billion known-vulnerable—and avoidable—**Java dependencies** are downloaded each month. The problem is also getting worse. Sonatype, for instance, discovered a 742% average yearly increase in software supply chain attacks since 2019. And all signs indicate this figure will increase as open source usage grows.

Sonatype discovered a 742% average yearly increase in software supply chain attacks since 2019.

This is a troubling development, considering the rapid pace of software adoption among all industries. As companies continue to go through digital transformations and become more dependent on software, technical teams must do a better job of discovering and closing vulnerabilities to prevent dangerous and costly cyber incidents from taking place.

Let's explore the growing demand for software supply chain management with end-to-end visibility, transparency, and tracking across all levels. By adopting supply chain security best practices, companies can stay on top of evolving threats and prevent them from impacting operations.

Read on for a breakdown of the following issues that are central to software supply chain security:

- The meaning of software supply chain management.
- The growing need for software supply chain management.
- Why adversaries are attacking the supply chain.
- The evolution of software supply chain attacks.
- An explanation of dependency confusion and why it's a threat.
- How companies should approach supply chain responsibility and management.

What Is Software Supply Chain Management?

Software supply chain management is a strategy that aims to protect digital supply chains from vulnerabilities and help companies produce stronger and more reliable products. It's similar to the concept of supply chain management for physical products but with a digital focus.

This strategy involves monitoring and tracking activities across all stages of the software development lifecycle, including components like hardware, operating systems, and cloud services. It also involves the sources where code originates, such as GitHub repositories and other open source projects.

The Growing Need for Software Supply Chain Management

The main reason why companies need software supply chain management is that poor configuration and sourcing practices can lead to dangerous outcomes— especially when involving mission-critical infrastructure, financial institutions, and healthcare facilities.

Currently, there's a software supply chain crisis, as companies lack deep visibility and control over the various open source components that make up their digital services. And as a result, customers who depend on software are at risk.

Of course, software is one of the many industries to face supply chain challenges.

- Boeing sent shockwaves through the airline industry in 2013 when multiple **787 planes caught on fire** while charging at their gates due to battery problems.
- Chevrolet **issued a total recall** for the Chevy Cobalt in 2014 when the model's ignition switches caused some drivers to lose power while driving.
- The Centers for Disease Control and Prevention (CDC) **issued a warning in 2018**, instructing consumers not to eat any romaine lettuce from the Yuma, Arizona, growing region due to an outbreak of E. coli.

All the above incidents were resolved with varying competency levels, efficiency, and total damage. What's important to note is that the automobile, food, and airline industries are held to a high standard of excellence. Consumers and regulators expect sound manufacturing practices and have no tolerance for engineering or production mistakes that can lead to fatalities or hospitalizations. As a result, companies place great emphasis on supply chain and production quality.

Supply chain errors happen much more frequently in the software industry than in other industries. However, software companies are held to a different standard of excellence. As a result, dangerous errors and vulnerabilities persist and regularly make their way into production.

This is becoming a more significant issue because companies and industries are becoming increasingly digital and interconnected. For example, automobiles, manufacturing facilities, and smart cities increasingly rely on software with open source components to operate. At the same time, 78% of CIOs say that scaling AI will be their **top priority for data strategy** by 2025. And many organizations view **open source** as a pathway to unlocking the power and benefits of AI and enabling artificial general intelligence (AGI), or the next phase of AI where computers meet and exceed human intelligence.

Organizations that fail to address underlying quality and security issues in their code risk suffering devastating attacks that could cause significant physical and financial harm. Companies need to catch up regarding open source security. In fact, only 50% of enterprise developers currently have processes in place to evaluate, approve, or standardize a new dependency. What's more, only 37% use automated tools for tracking dependencies, managing dependencies, and ensuring policy compliance.

Automobiles, manufacturing facilities, and smart cities are increasingly relying on software with open source components to operate.

Sonatype's latest **State of the Software Supply Chain report** reveals the average Java application now contains 148 dependencies—20 more than last year. The average Java project also updates ten times each year, meaning that developers must track intelligence on almost 1,500 dependency changes per year and per application they work on.

Why Adversaries Are Attacking the Software Supply Chain

In most cases, financial gain is the **primary motivation** for a software supply chain attack. Threat actors typically want the fastest, easiest, and safest path to profits. On a global scale, **cybercrime will cost** the world $10.5 trillion by 2025.

But despite the enormous opportunity at hand for cybercriminals, it's becoming harder to launch traditional attacks against software and applications. Companies are increasingly prioritizing cybersecurity by deploying real-time threat monitoring solutions, increasing employee cyber awareness training, and switching from monolithic architectures to container-based architectures that make it harder to achieve lateral movement across the organization.

As a result, threat actors are starting to switch gears and focus on penetrating vulnerable software supply chains. Instead of going through the hassle of attacking an application, hackers can instead target and distribute application components that can infect larger volumes of applications and end users at once.

In other words, the software supply chain provides a simpler point of entry for cybercriminals, with a wider target base—and the potential to rake in more money than traditional attacks.

The Evolution of Software Supply Chain Attacks

Software supply chain attacks are not a new challenge for security teams. They have been occurring and evolving over the last decade, with threat actors continuously innovating and looking for new opportunities.

Here's a general look at the evolution of software supply chain attacks, broken down into three phases.

Phase 1: Exploiting Open Source Vulnerabilities

The first phase involves exploiting existing open source vulnerabilities in code. With this strategy, threat actors aim to attack vulnerabilities faster than companies can remediate them.

One of the most notorious open source software vulnerability exploits is the first **Apache Struts framework** vulnerability of 2013. During this incident, attackers could gain remote access to any Struts application. The exploit required no authentication and allowed hackers to access information, make modifications, and disrupt service. Numerous companies, including large financial institutions, were impacted.

Two other high-profile exploits include the **Heartbleed** and **Shellshock** attacks, which happened in the following year. Heartbleed was a security bug in the OpenSSL library commonly used for Transport Layer Security (TLS). At the same time, Shellshock is a bug in the Bash CLI that went undiscovered for 30 years until 2014.

These were two devastating attacks. Heartbleed exposed roughly **half a million secure servers** to password theft and was also used to attack a hospital. The **number of Shellshock attacks** also topped one billion within the first week of its existence—and the vulnerability remains unresolved today.

It's also worth mentioning the Apache Commons Collection vulnerability of 2015. Commons Collections is a popular utility library in Java and exists in just about every Java application. This massive vulnerability impacted 78% of the 23.4 million downloads in 2016. Hackers also used this vulnerability to launch a ransomware attack at **Hollywood Presbyterian Medical Center**, knocking the hospital offline.

Phase 2: Creating Opportunities in the Supply Chain

Open source attacks are still taking place on a large scale, with hackers frequently discovering new vulnerabilities and trying to exploit them before companies can address them. However, attackers are now also leveraging more sophisticated ways to exploit the software supply chain.

In the second phase of this evolution, attackers are creating opportunities in the supply chain. Many are switching gears and moving away from going into actual applications in favor of trying to attack open source publishers.

Attackers are now also leveraging more sophisticated ways to exploit the software supply chain.

One of the first instances occurred in 2017, with hackers launching typosquatting attacks in npm and Python to steal open source publisher credentials and publish harmful code as the actual publishers. Many subsequent attacks followed soon after, as word of this successful strategy began to spread.

Phase 3: Attacking Developers and Development Infrastructure

In the latest evolution of software supply chain attacks, threat actors are beginning to attack developers and development infrastructure.

One of the first examples of this occurred in 2018 when a hacker exploited an unpatched version of Jenkins and used it to mine for Bitcoin. At the time, this was one of the biggest malicious mining operations in existence.

In a more recent example involving Jenkins, attackers were able to leverage a system vulnerability to hack into security provider **Verkada's** network. Attackers could access the company's development infrastructure and move laterally within the organization to access private data.

Yet another example is the Codcov incident, where attackers could exploit a vulnerability using a Docker container and deploy the container downstream, which became part of the supply chain of other customers and led to additional attacks.

As you can see, hackers are no longer merely trying to break into applications. Instead, they are using attacks as entry points into organizations. Application security professionals must therefore focus on defending the development perimeter because it's now a key part of the operational perimeter.

Dependency Confusion: A New and Evolving Threat

Dependency confusion is a new attack involving determining a company's internal components and registering its name in public registries.

In this type of attack, an actor will visit a repository and upload a component with the same name that a company is using internally and trick the build tools into selecting their hacked version.

At first, dependency confusion began as a white hat research discovery when researcher Alex Birsan was able to get targets to **automatically install his code** at Microsoft, Apple, and 33 other companies. The research led to a total bounty payout of $130,000.

However, the attack also opened the floodgates for copycat white hat hackers and bad actors who soon began publishing **malicious squatted dependency confusion packages** to the npm ecosystem and targeting companies like Lyft, Amazon, Zillow, and Slack.

Companies must now exercise caution when accessing packages from repositories and take advanced measures to prevent malicious code from making its way into software systems.

Why Log4Shell is an Eye-Opening Security Incident

Despite all these examples, many companies still need to learn about software supply chain threats, which is why they continue downloading harmful components and installing them into software and applications without considering the implications. And when issues do arise, business leaders often need help with how to proceed.

This was made abundantly clear following the discovery of **Log4Shell**, a widespread vulnerability in a Java software component called Log4j that primarily impacts servers and web-based applications. The Apache Foundation publicly revealed the Log4Shell vulnerability in December 2021.

According to Mandiant, China-backed APT41 began exploiting Log4Shell within hours of the Apache announcement, compromising two U.S. state government networks and other high-profile targets in the telecoms and insurance industries.

Not only was the vulnerability widespread, but it was also very easy to exploit—meaning all businesses using Log4j should have immediately remediated the issue by upgrading to the latest version. However, downloads of the unpatched version of Log4j containing the Log4Shell vulnerability continued long after the announcement.

Log4Shell was a clear indicator that businesses are still failing to understand the severity of open source vulnerabilities.

This was a clear indicator that businesses still need to understand the severity of open source vulnerabilities, despite all the work that goes into closing them and spreading awareness. To improve software supply chain security, companies must do more to educate leaders and developers about the issue and create policies to discover and remediate vulnerabilities when they arise.

Who Should Take Responsibility for Open Source Security?

The Log4Shell vulnerability also brings to light several important questions about the state of open source security and responsibility.

Right now, a debate is raging about how to prevent open source security issues from impacting companies and consumers. For example, some industry professionals are calling for open source funding to compensate volunteers for participating in closing open source vulnerabilities. In addition, there's growing concern that policy and even federal intervention are necessary to protect open source security.

With that in mind, let's examine some general proposals for addressing open source security.

Improving Memory Safety

Memory safety issues continue to plague open source projects. As such, some experts call on projects to use languages with fewer memory safety issues, like Java or Rust instead of C++. This is a viable option, but it's a long-term goal that will take years to achieve.

Increasing Security Education

Schools and universities must better educate developers about application security and prepare them for real-world threats and vulnerabilities. However, this requires buy-in from educators and will take years before seeing any meaningful results.

Expanding Code Audits

This approach involves providing more private funding to create more investigations and audits and encourage developers to review code for vulnerabilities. This plan could work, but developers must have deep knowledge of specific code.

Creating Better Scanning Tools

Currently, there's a shortage of scanning tools for open source projects. Creating more free scanning tools would help increase visibility for developers. Again, this is not a short-term solution and will take time to execute.

These are just a few of the numerous suggestions that the software industry is considering. Other options on the table include conducting more risk assessments, improving digital signatures for components, optimizing incident response, and creating tighter requirements for software bills of materials (SBOMs). The list goes on.

The problem with most of these suggestions is that they shift responsibility from software manufacturers to open source suppliers and community members. Companies must take better ownership of supply chain security and do a better job tracking components and building solid software versions. Until companies begin prioritizing software supply chain security, open source vulnerabilities will continue to be a significant threat.

At the end of the day, companies must take better ownership of supply chain security and do a better job of tracking components and building solid software versions.

How Companies Should Approach Supply Chain Responsibility and Management

Instead of waiting for the industry to change with more funding, regulation, and oversight, companies must take action and accept full responsibility for their software supply chains.

Executing the following steps can enhance security, enable faster resolutions, and lead to better and more resilient software.

Make Better Choices

Companies must carefully consider the components they use in their software versions and use monitoring and automation tools to discover vulnerabilities and prevent hacked components from making their way into their environments. In addition, developers must carefully track versions and maintain meticulous records when iterating and making changes to their code.

Maintain an Organizational Software Bill of Materials (SBOM)

Companies must also maintain comprehensive SBOMs to locate vulnerabilities when they arise and perform effective maintenance. SBOMs are critical for ensuring accountability and notifying customers about security issues. This should be a requirement for all organizations that are using open source components.

Have a Managed Response

It's impossible to prevent vulnerabilities when relying on open source components entirely. However, having a managed response in place can help to align efforts and remediate issues more efficiently. This strategy can reduce panic and prevent decision-makers from making costly mistakes that could lead to further reputational harm.

You Have a Supply Chain—Even If You Don't Manage It

The key takeaway here is that any business relying on third-party, open source components has a software supply chain that requires ongoing management.

Managing software supply chains plays a massive part in an organization's overall security strategy. When ignored, they risk leaving the door open for third-party threat actors to access sensitive data and private resources.

Make no mistake: attackers are actively looking to exploit software supply chains by attacking upstream. Left unmanaged software supply chains are some of the top targets for cybercriminals in 2023, and they present a quick and easy way for hackers to generate income.

The good news is that finding and addressing open source vulnerabilities is easier than ever.

Sonatype provides an industry-leading open source security and dependency management platform that automatically finds and fixes open source vulnerabilities across every stage of the software development lifecycle, saving time and reducing risk.

To experience how Sonatype can enhance open source security and help provide greater transparency into your software supply chains, **request a demo today**.

sonatype

Sonatype is the software supply chain management company. We empower developers and security professionals with intelligent tools to innovate more securely at scale. Our platform addresses every element of an organization's entire software development life cycle, including third-party open source code, first-party source code, infrastructure as code, and containerized code. Sonatype identifies critical security vulnerabilities and code quality issues and reports results directly to developers when they can most effectively fix them. This helps organizations develop consistently high-quality, secure software which fully meets their business needs and those of their end-customers and partners. More than 2,000 organizations, including 70% of the Fortune 100, and 15 million software developers already rely on our tools and guidance to help them deliver and maintain exceptional and secure software. For more information, please visit **Sonatype.com**, or connect with us on **Facebook**, **Twitter**, or **LinkedIn**.

Headquarters	European Office	APAC Office	Sonatype Inc.
8161 Maple Lawn Blvd, Suite 250	168 Shoreditch High St, 5th Fl	60 Martin Place, Level 1	www.sonatype.com
Fulton, MD 20759	London E1 6JE	Sydney 2000, NSW	Copyright 2023
USA • 1.877.866.2836	United Kingdom	Australia	All Rights Reserved.