



WhiteSource

# THE DEVIL IN THE DETAILS: The Importance of SBOMs in Protecting the Software Supply Chain

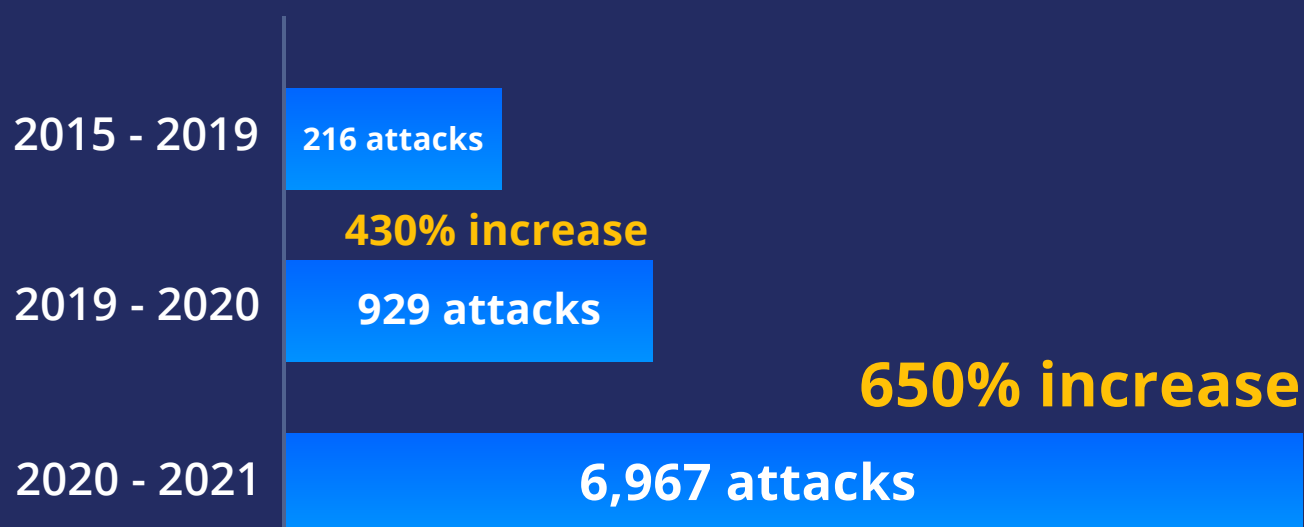




When security researchers discovered in December 2020 that attackers had trojanized software updates in a SolarWinds application, it was a rude introduction into software supply chain attacks for more than 18,000 businesses and governmental agencies.

While SolarWinds is the largest and best known software supply chain attack thus far, it's unfortunately not a unique occurrence. The reality is that attacks against the software supply chain are one of the most pervasive threats that companies face, with attackers launching almost 7,000 software supply chain attacks in just the past year, as bad actors look for ways to steal data, corrupt targeted systems and gain access to other parts of the network through lateral movement.

**Figure 1: Increase in Software Supply Attack Frequency**



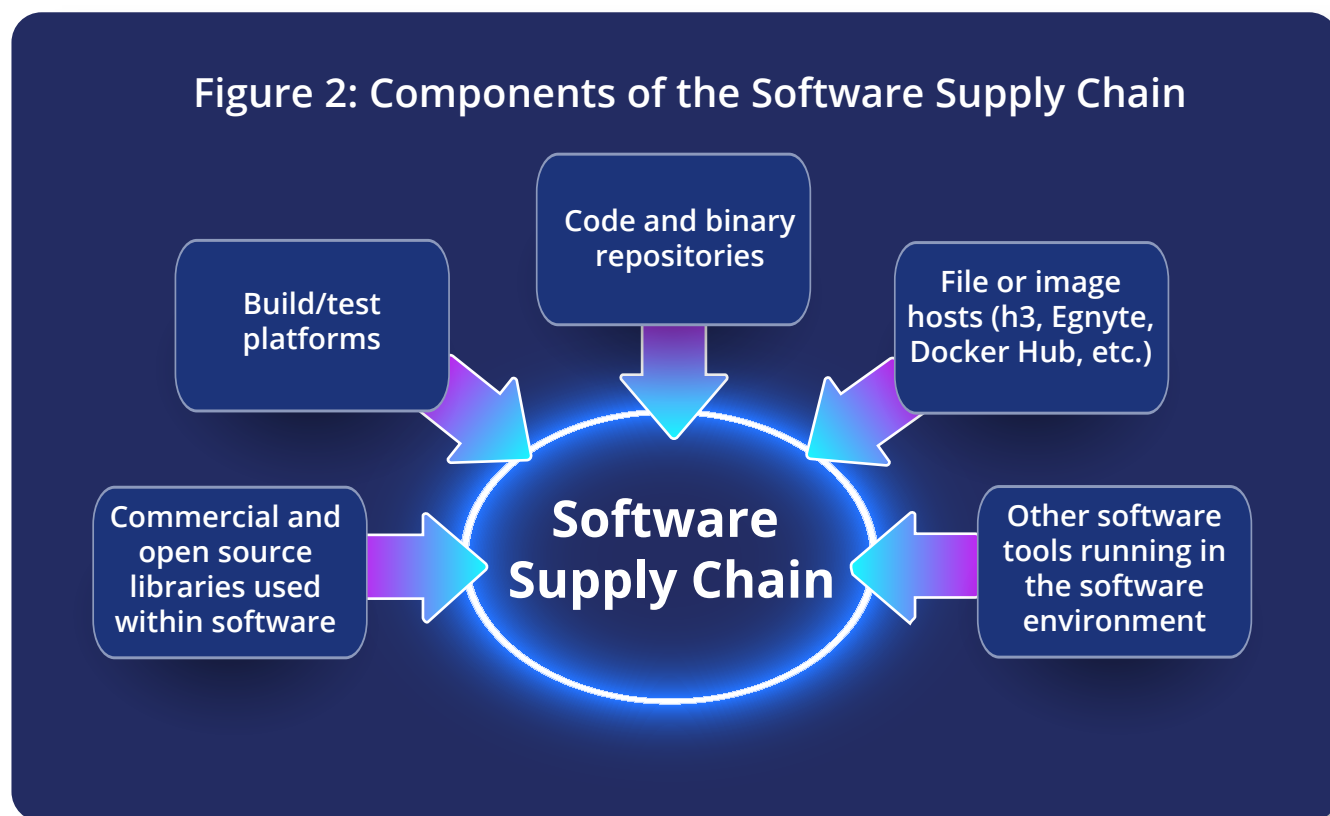
Software supply chain attacks occur when a bad actor infiltrates a software vendor's network, employing malicious code to compromise the software before the vendor sends it to a customer. The compromised software then compromises the customer's data or system. Newly acquired software may be compromised from the outset, or a compromise can occur through a patch or hotfix. Such attacks affect all users of the compromised software and can have widespread consequences for government, critical infrastructure and private sector software customers.





According to the European Union Agency for Cybersecurity (ENISA), almost 60 percent of supply chain attacks are aimed at gaining access to data – including personal data and intellectual property – and around 16 percent of attacks are an attempt to gain access to people. It should also be noted that in 66 percent of incidents, attackers focused on suppliers' code in order to further compromise targeted customers.

The software supply chain is made up of everything that goes into or affects code from development all the way until it's deployed into production. It includes code, binaries and other components as well as and includes information about who wrote it, when it was contributed, how it's been reviewed for security issues, known vulnerabilities, supported versions, and license information – basically everything that touches it at any point.



As shown in Figure 2, the software supply chain consists of every non-organic service or piece of software that's used to build software. Importantly, this includes components and packages that are open source – a massive undertaking given the use of open source code bases has increased from 36 percent in 2015 to 75 percent in 2020.

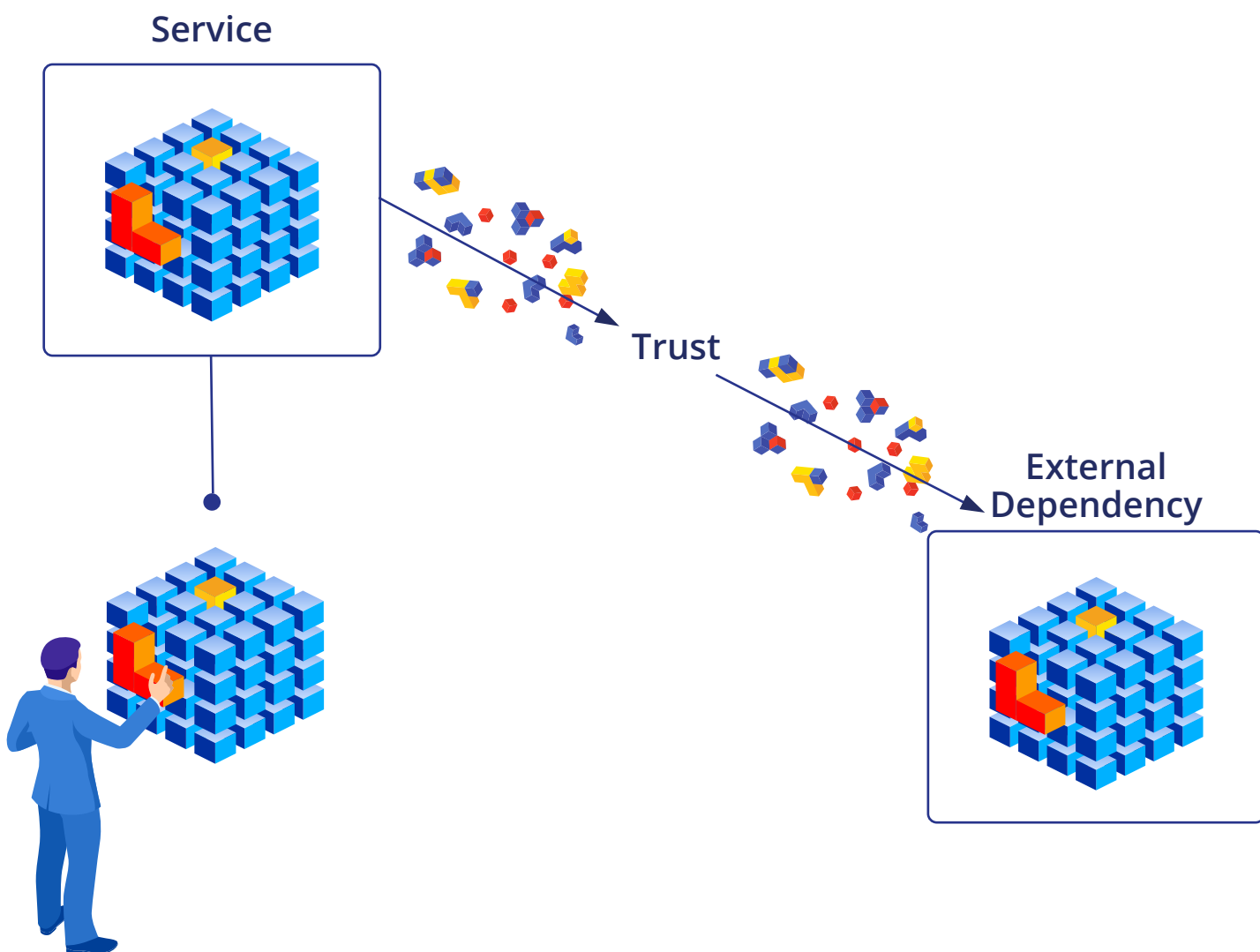




Open source is extremely attractive to attackers, who implant malware directly into open source projects to infiltrate the software supply chain. In the past, software supply chain exploits were used against publicly disclosed open source vulnerabilities that were left unpatched. But instead of waiting for disclosures to pursue an exploit, attackers now are choosing to inject new vulnerabilities into open source projects that feed the global supply chain – and then exploit those vulnerabilities before they are discovered.

The growth in software supply chain attacks has – not surprisingly – resulted in a firestorm of activity on the part of regulators and governmental organizations to determine how such attacks can be prevented.

A key facet for all of the regulatory agencies involved with securing the software supply chain is the demand for a software bill of materials (SBOM), a formal record that contains the details and supply chain relationships of various components that are used to build software.





# Timeline Of Regulatory Efforts To Address Software Supply Chain Attacks

**1 February 2021**  
President Biden issues an Executive Order (EO) laying out changes to secure all supply chains, including software, calling for the secretaries of Commerce and Homeland Security to report on the security and integrity of critical information and communications technology software supply chains.

**2 February 2021**  
The CIO for the Department of Defense rolls out a new reference for Zero Trust Architecture (ZTA), including a section centered on protecting applications and software supply chains.

**3 April 2021**  
The National Institute of Standards and Technology (NIST) and the Cybersecurity and Infrastructure Security Agency (CISA) release Defending Against Software Supply Chain Attacks, acknowledging that software compromised in supply chain attacks could have widespread consequences. It outlines safeguards to secure the software supply chain, including:

- Developing a written program to address software supply chain risk
- Inventorying organizational reliance on external software and code
- Assessing risk from vendors and adopting contractual and other safeguards
- Monitoring threats and vulnerabilities to the software supply chain

**4 May 2021**  
Biden signs a second EO for software supply chains as part of a critical look at the nation's cybersecurity posture. It requires NIST to create guidelines to secure the software supply chain, including the need for all critical software to include a software bill of materials (SBOM).

**5 June 2021**  
NIST defines critical software as having at least one of these attributes:

- Is designed to run with elevated privilege or manage privileges
- Has direct or privileged access to networking or computing resources
- Is designed to control access to data or operational technology
- Performs a function critical to trust
- Operates outside of normal trust boundaries with privileged access

**6 July 2021**  
NIST publishes security measures for critical software use and minimum standards for vendors' testing of their software source code.

**7 July 2021**  
The National Telecommunications and Information Administration (NTIA) releases the minimum elements needed for an SBOM to improve transparency in software supply chains for both technology vendors and government customers.







## What Are SBOMs?

The NTIA describes SBOMs as a nested inventory – a list of ingredients that make up software components – which provides those who produce, purchase and operate software with information that enhances their understanding of the supply chain.

It is a formal record that contains the details and supply chain relationships of various components used in building software, including libraries and modules. SBOMs are a formal, machine-readable inventory of software components and dependencies that contain information about those components and their hierarchical relationships.

While SBOMs don't directly improve the security of products, the resulting transparency from creating SBOMs allows those who produce, purchase and operate software to better track and fix known and newly emerged vulnerabilities.

SBOMs can include open source or proprietary software, and they can be openly accessible or proprietary. At a minimum, SBOMs should include:

- ✓ **Data fields:** Tracks and documents baseline information about each component. This includes dependencies and hierarchies of components, such as open source direct and transitive libraries.
- ✓ **Automation support:** Allows for scaling across the software ecosystem through automatic generation and machine readability in either SPDX, CycloneDX or SWID formats. Any changes to any component, such as a new library version being added, should automatically generate updates to the SBOM.
- ✓ **Practices and processes:** Defines the operations of SBOM requests, generation and use.
- ✓ **Baseline component information:** Includes open source and commercial software component information.

For producers and operators of software, this means tracking inventory, understanding relationships between inventory items, tracking vulnerability and license management, and ensuring that components are up to date. When new vulnerabilities are identified, a quick and accurate assessment of the risk is to be shared across dependencies within the software ecosystem. Doing so improves both vulnerability identification and the speed of response. Overall, SBOMs should provide producers and operators of software with greater efficiency and effectiveness through visibility, which in turn enables prioritization and better management of risk.










To better illustrate why so much attention is being placed on SBOMs, consider two scenarios.

## Scenario 1: SBOM Done Incorrectly

There are many tools on the market that will technically create an inventory of open source components; however, the majority of those tools are not suitable for SBOM for a number of reasons, including:

-  Many tools identify vulnerabilities but don't accompany such identification with a clear path to address those risks. Each time a vulnerability is identified, the development team must stop to research a fix, significantly slowing product development.
-  Likewise, many of the tools only cover certain languages or package managers, resulting in an incomplete SBOM. Partial SBOMs don't meet mandated requirements, and they expose the producer to risks.
-  The mandate calls for SBOMs to be automatically generated in machine-readable format. Tools not built for full automation won't meet the requirements – and they'll slow down product releases.
-  Tools that lack the ability to scale across the software ecosystem, particularly across organizational boundaries, create an operational bottleneck and slow down the release process.
-  Most tools have a high false-positive rate, including misidentification of libraries and transitive dependencies. As such, these tools create SBOMs that expose the enterprise to additional risks from vulnerable components that effectively can't be fixed.










## Scenario 2: SBOM Done Correctly

In a remediation-centric approach, identification of vulnerable components is fully integrated into the cycle of addressing those risks. The focus is on helping developers remediate issues quickly and easily, while also keeping code up to date.

This starts at the sourcing stage to help avoid problematic components and continues with the identification and recommendation of component upgrades as they are released. Unreachable vulnerabilities are removed, with each vulnerability being properly prioritized and matched to a best fix.

The resulting SBOM reflects the health of the product, and the process is used to continuously identify and address supply chain risks. As such, it speeds up the development process and becomes a competitive advantage.

When done correctly, SBOMs provide the following:

-  Common language or framework of understanding for software assets (also known as the standard identifier)
-  All aspects of the SBOM are automated
-  The ability to identify and manage software supply risk, including the ability to detect and mitigate risk in downstream components when an upstream component is compromised
-  Improved security that avoids and addresses security risks before a software component is chosen for use
-  Improved supply chain risk management

## How Can WhiteSource Help?

As the number of software supply chain attacks continues to increase, regulators will respond by pressuring producers and operators of software to ensure that they're implementing SBOMs and other measures for remediation and protection. Get in touch today to learn more about how Whitesource is helping customers implement a remediation-centric approach to help you protect the software supply chain.

