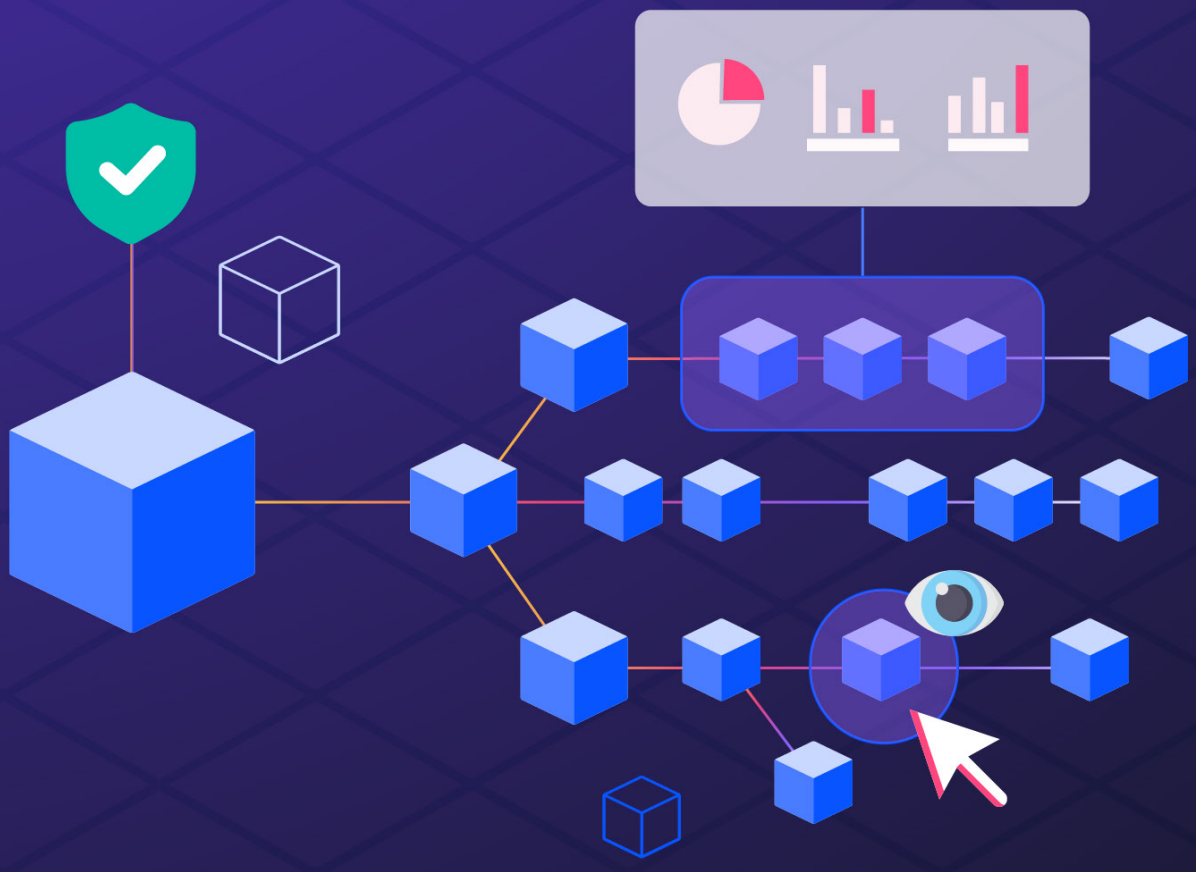
ActiveState

SOFTWARE SUPPLY CHAIN SECURITY BUYERS GUIDE **FOR IMPORT TOOLS**



Executive Summary

The software supply chain isn't a new concept, but in a post-Solarwinds world, it's become the focal point of efforts to improve cybersecurity in general, and the security of software development in particular. The market result has been a gold rush as traditional software security vendors have quickly positioned themselves as supply chain security vendors. Making heads or tails of their claims can be confusing.

The process of importing third-party code into your organization is arguably the weakest link in the software supply chain since it involves trusting thousands of open source developers with whom you have no relationship.

Software supply chain risks increase with:

- The number of open source languages/ecosystems in your development stack
- The lack of key best practices present in your import routine
- The fewer codebase updates you perform

A number of traditional and emerging tools discussed in this guide can help mitigate many of the threats inherent in your software supply chain.

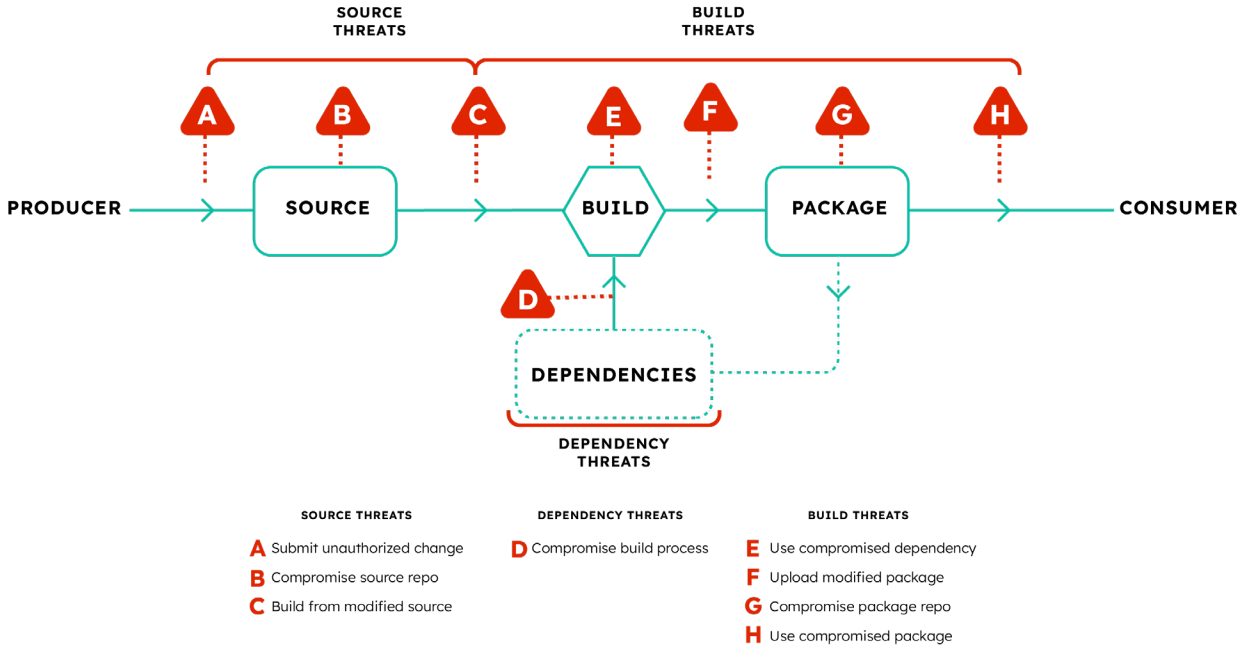
Introduction

Organizations are increasingly concerned with the security of their software supply chain, but often have trouble navigating the ever-expanding labyrinth of open source and proprietary software solutions that claim to help. While solutions exist across the entire software supply chain, this section of ActiveState's Buyers Guide to Securing the Software Supply Chain focuses specifically on the tools used to secure the process whereby third-party code is brought into the organization.

The software supply chain extends from:

- **Imported Code -** any open source packages, code snippets, tools, or other third-party software brought into the organization in order to streamline the software development process.
- **Build Process –** the process of compiling, building and/or packaging code, usually via an automated system that also executes tests on built artifacts.
- **Use/Deploy –** the process of working with, testing and running built artifacts in dev, test and production.

Hackers are increasingly targeting software development systems, open source resources and software build pipelines in order to gain economies of scale when it comes to compromising software supply chains. A single compromised software artifact in a popular application can potentially compromise hundreds or even thousands of downstream customers. It's one of the key reasons that software supply chains are increasingly under attack, with 2023 featuring twice as many attacks as the past 3 years combined, despite the fact that 2019-2022 saw a 742% increase[1] in software supply chain attacks, year on year.



1. Sonatype 9th Annual State of the Software Supply Chain

As a result, Gartner is predicting that “By 2025, 45% of organizations worldwide will have experienced attacks on their software supply chains, a three-fold increase from 2021[2]”. Online marketplace vendor Capterra confirms the trajectory, citing the fact that “three-fifths (61%) of US businesses have been directly impacted by a software supply chain threat” in 2022, and as a result, more than 50% of organizations have lost trust in legacy vendors due to supply chain attacks

> 74% of IT pros believe technologies like static and dynamic application security testing [SAST & DAST] are important, but feel that those technologies aren’t enough to protect them from supply chain threats
>
> – ReversingLabs

With IBM reporting that the average cost of a supply chain attack has hit $4.45M, requiring an average of 26 days to identify and contain, it’s no wonder that software supply chain security tools budgets have become an increasingly large part of the budget for security-conscious organizations.

2. Gartner report: How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks

Securing the Weakest Link in the Supply Chain

The import process is often seen as the weakest link in the chain since it requires trusting hundreds or even thousands of open source authors with whom the organization has no relationship. After all, the open nature of open source ecosystems allows anyone to publish anything without a rigorous process to eliminate threats prior to publication. As a result, each organization must create their own process, implement with appropriate tools and solutions, to ensure the integrity and security of the code they import.

First, you'll need to get a grasp on the breadth, depth and change associated with your software supply chain:

- **Breadth –** most organizations work with multiple open source languages, and import their code from more than one public repository. Because there are no industry-wide standards in place today, each language and repository may require its own solution.
- **Depth –** there is a large set of supply chain security & integrity best practices that can help, but only the largest enterprises can hope to implement and maintain them all. You'll need to find the biggest bang for your buck.
- **Change –** most software artifacts have a short shelf life before bugs, vulnerabilities, incompatibility with newer systems and other issues crop up. Correcting all these issues is difficult for all but the most dedicated organizations to do in a timely manner. Automated solutions can help.

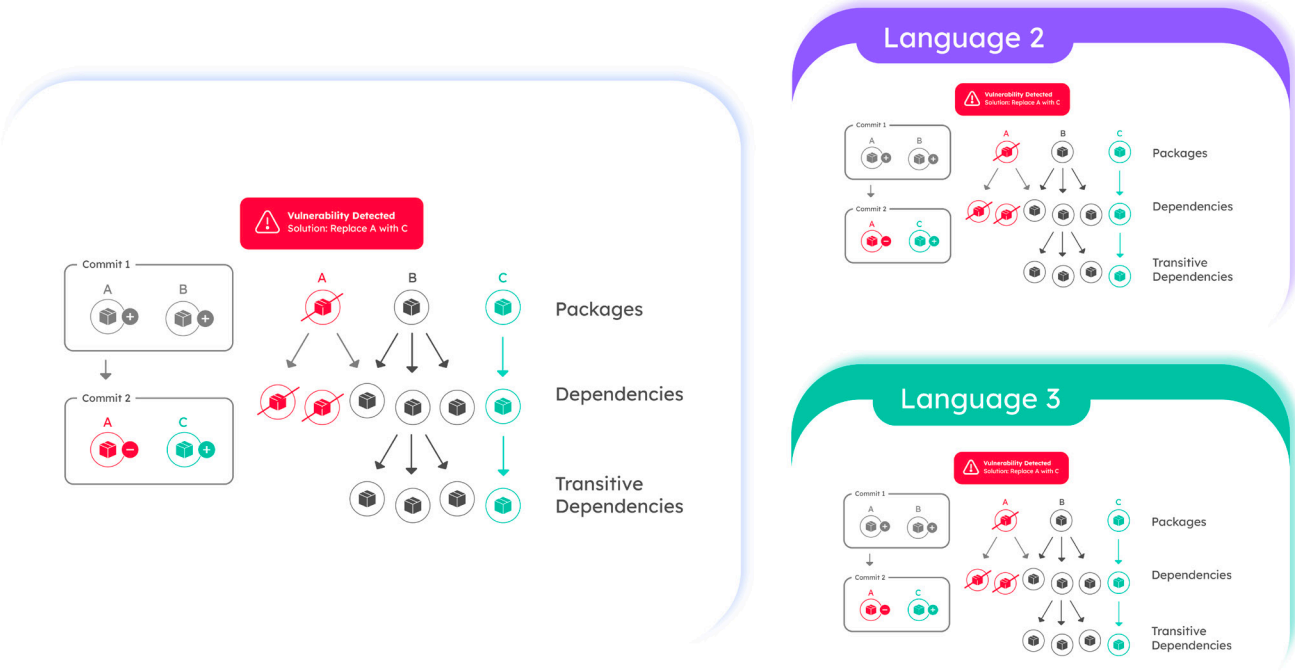


Figure 1:
multiple ecosystems and multiple dependencies multiply the effects of change across the import process

Breadth of the Software Supply Chain

The breadth of the software supply chain expands every time you add a new open source ecosystem to your development stack.While issues remain consistent, how they are expressed in the language may differ greatly, requiring different sets of tools to deal with:

- **Vulnerabilities** - security issues in imported packages can pose significant risk.
  - » Software Composition Analysis (SCA) tools can be used here to generate a manifest of all open source components, along with their open source license(s) and known security vulnerabilities.
- **Malware** - packages containing malicious code can compromise your development environment or software solution.
  - » Code scanners can help identify threats such as typosquatted packages, malware, dependency confusion, and more. Scanners come in many forms, from open source point solutions to commercially sold security platforms. Collectively, they can help you erect a powerful, in-depth defense, but even individually they can provide an effective solution depending on your need.

While numerous open source and commercial solutions exist to help deal with each of these issues, keep in mind that:

- Ecosystem-specific tools tend to be more robust, but will significantly increase costs to implement and maintain if you work with more than one language.
- Commercial solutions tend to be more comprehensive and expensive, but may be more cost-effective in the long run than trying to integrate and maintain multiple, best-in-class open source solutions.

Depth of the Software Supply Chain

The depth of the software supply chain increases with each set of best practices you implement to counter a potential threat vector. Key best practices include:

- **Verifying Provenance -** the origin of imported code should always be verified on import to ensure it has followed best practices when it was developed and built/packaged for distribution.
  - » Software attestations can be used here to help understand the risk of incorporating imported code into a codebase. TestifySec Witness provides a framework for automating, normalizing, and verifying software attestations.
- **Dependency Vendoring -** the best way to ensure the security and integrity of imported code is to vendor the source code of all required open source dependencies and build them yourself. Unfortunately, this means you are now responsible for patching and maintaining all of that third-party code, which can overwhelm many organizations.
  - » Automated dependency vendoring tools can help here by automatically managing dependencies, building them all from source code, and then facilitating updates when components become outdated or vulnerable.

- **Quarantining -** imported software artifacts should be quarantined in a repository during investigation/ scanning. You may also want to consider a separate repository for vulnerability identification and a further one for "ready to use" artifacts.
  - » Artifact repositories like Sonatype Nexus, JFrog Artifactory, or a similar solution can be used to quarantine prebuilt artifacts.

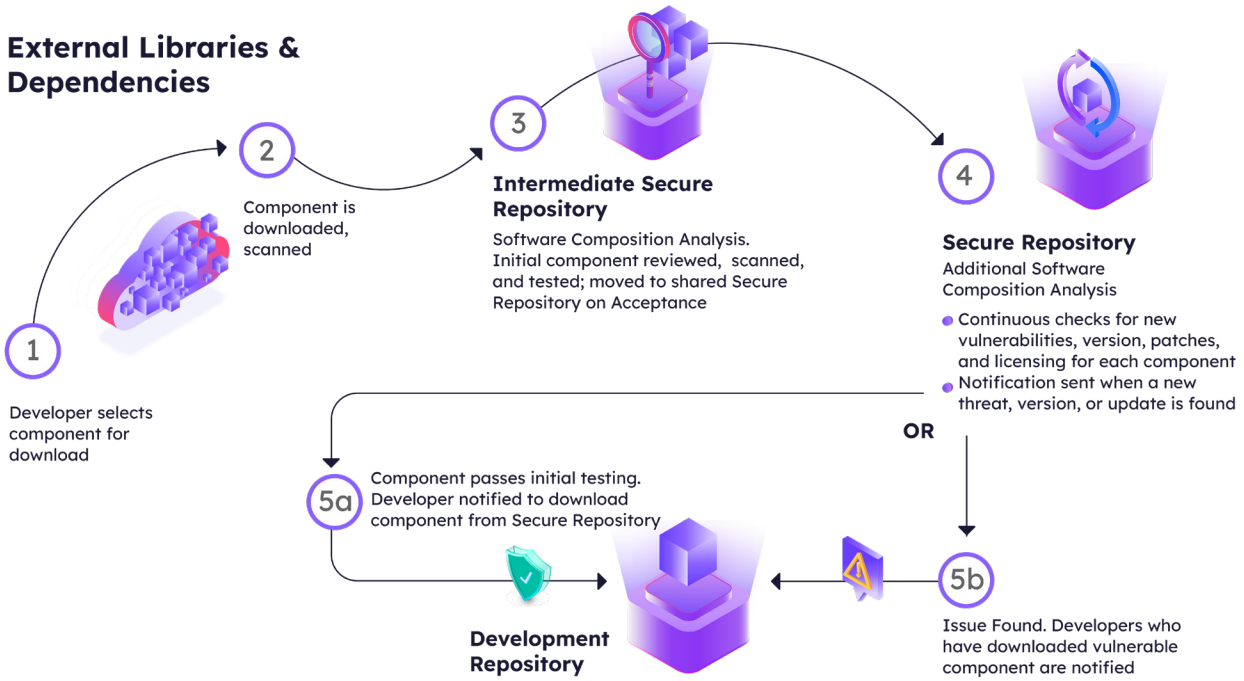


Figure 2:
Quarantining third party components during the import process

Change in the Software Supply Chain

While the rate of change in the software supply chain can be managed using many of the tools already discussed, too many organizations are still reluctant[3] to update their codebase for a number of reasons, including:

- **Time & Resources -** developers dedicated to fixing issues are developers not available to create new features.
  - » Many of the aforementioned SCA tools can help decrease the time developers spend investigating issues by providing auto-updates.

- **Breaking Changes -** upgrading a package has a cascading effect on its dependencies, requiring multiple updates to your runtime environment which can end up breaking the build, or worse: landing you in dependency hell.
  - » Look for automated dependency management tools here to help avoid dependency hell, while streamlining the upgrade process.

3. Veracode's State of Software Security v11: Open Source Edition

As the rate of discovered vulnerabilities continues to escalate year over year, organizations need to become more proactive about not only updating their codebases, but ensuring known vulnerable components aren't incorporated in the first place.

> 2.1 billion OSS downloads with known vulnerabilities in 2023 could have been avoided because a better, fixed version was available."
>
> – Sonatype

Conclusions

It seems like every vendor is now a software supply chain security vendor, causing confusion in the marketplace. But avoiding market confusion is not an option since it only makes businesses increasingly susceptible to the threat of ransomware, malware, and other security risks. Organizations need tangible solutions that can help them address threats when they first enter the development process – on import – before they can do damage.

Following Biden's Executive Order[4] on improving cybersecurity, many are still trying to figure out what that looks like for their own organization. As software supply chain attacks hit unprecedented levels, it may be time to start rethinking whether downloading prebuilt open source software makes sense anymore. Organizations that don't vendor their dependencies and build them from source code can only play catch up with supply chain security via traditional AppSec tools – tools that are often working against an incomplete set of dependencies since their dependency graph wasn't generated at build time.

Dependency vendoring has always been a non-starter for all but the biggest enterprises, but emerging automated solutions offer a cost-effective alternative. This will be the focus of the second part of ActiveState's *Software Supply Chain Security Buyers Guide for Build Tools*.

4. Executive Order on Improving the Nation's Cybersecurity, MAY 12, 2021

ActiveState

Secure open source integration

www.activestate.com
Toll-free in NA: 1-866.631.4581
solutions@activestate.com