

Next-Gen Software Composition Analysis: Pipeline Composition Analysis

TOP 10 KEY REQUIREMENTS FOR REDUCING THE RISK OF VULNERABLE DEPENDENCIES



01 — Detection of All Pipeline Dependencies, Not Just Application Code

Only scanning your application code for vulnerable dependencies is not enough to protect against modern threats like software supply chain attacks. You need to **scan your entire SDLC** for **many types** of vulnerable dependencies because they exist throughout your entire pipeline, including:

- Dev tools (e.g., Jenkins)
- Dev tool plugins (e.g., Jenkins Plugins)
- Build modules (e.g., Github Actions)
- Build module dependencies (e.g., Github Actions libraries)
- Infrastructure as Code (IaC) dependencies

Traditional software composition analysis (SCA) solutions have no visibility into these pipeline dependencies and cannot determine whether they are vulnerable. This is the job of Pipeline Composition Analysis (PCA). Select a PCA solution that can find vulnerabilities in these other types of dependencies that occur throughout your entire pipeline.

02 — Vulnerable Dependency Deployment Location

When exploits like Log4Shell arise, removing all instances from all production environments where it could be exploited, in a complete and timely manner, is essential. You not only need to be able to identify all vulnerabilities in your code base, but also every instance of that vulnerability that has been pushed to production systems and could potentially be exploited. Choose a solution that can identify and pinpoint vulnerability locations so that you can be secure knowing that every potential exploit has been remediated.

03 — Runtime Exploitability

Given the number of vulnerabilities that need to be remediated, you need a way to prioritize beyond CVSS scores so you can fix those with the highest risk first. Looking at whether a vulnerability is reachable in code is one way to do this. A better way, however, is to **identify whether a vulnerability is exploitable at runtime** based on what application components are in use in production environments.

04 — Remediation

The remediation advice provided by your SCA solution should give detailed information on the vulnerability itself as well as outline a path to a fix to save your developers' and security teams' time. Ensure you also have the ability to define automated policies and workflows based on security metrics and your own unique needs.

05 — Cloud Native Applications

Modern applications rely on Infrastructure as Code (IaC) to define run-time environments and are often deployed via containers. Choose an SCA solution that easily scans for and remediates IaC misconfigurations and security vulnerabilities found in containers.

06 — Pipeline Security & Governance

As development pipelines become more complex, managing security policies across the entire SDLC becomes more painful. Look for a solution that identifies pipeline tooling misconfigurations and implements best practices for security controls across the entire pipeline. Furthermore, choose a solution that identifies when pipeline security controls like SCA scanning are removed or disabled to prevent your tooling from being bypassed.

07 — SDLC & Pipeline Insight

Because vulnerable dependencies that impact your application can be found across the SDLC, you need to understand all SDLC tooling, such as activity, settings, configuration, users, to protect your application. Choose a solution that deeply understands your SDLC and brings that context to your SCA findings to improve accuracy.

08 — Correlate Risk Across AppSec Engines

Some SCA vendors can collect data from various engines, however very few are able to correlate it in meaningful ways. Look for a solution that correlates risks and threats across phases of the SDLC as well as all AppSec scanning tools. This helps your security team understand how risks relate to each other and how to best tackle security initiatives.

09 — Developer Workflow Integrations

Giving developers the tools to check for potential vulnerable dependencies before code is committed as part of a pull request is a key way to shift security left. To fully promote the adoption of a security tool like SCA, you need to give developers a platform that is intuitive to use and integrates with the existing workflows and tools they use every day. A good tool not only surfaces potential issues but also guides remediation by providing actionable fix guidance.

10 — Language & Package Manager Support

When choosing an SCA tool, ensure that it covers the programming languages and package managers that you use to build your applications. Focus on tools that provide support for your core development languages.

Pipeline Composition Analysis (PCA) vs. Legacy SCA

NEXT-GEN SOFTWARE COMPOSITION ANALYSIS



		PCA (Next-Gen SCA)	Legacy SCA
 Types of Dependencies Examined for Vulnerabilities	Code dependencies	✓	✓
	Dev tools (e.g., Jenkins)	✓	✗
	Dev tool plugins (e.g., Jenkins Plugins)	✓	✗
	Build modules (e.g., GitHub Actions, GitLab Runners)	✓	✗
	Build module introduced dependencies (e.g., dependencies used GitHub Actions)	✓	✗
	Infrastructure as Code introduced dependencies (e.g., scripts or packages introduced by IaC files)	✓	✗
 Scanning Locations and Techniques	Repos	✓	✓
	Pull requests and commits	✓	✓
	CLI	✓	✓
	Container registries	✓	✓
	IaC templates	✓	✓
 Pinpoints Vulnerable Dependency Locations	In code	✓	✓
	In production deployments	✓	✗
 Pipeline Analysis and Insight	Understand SDLC tooling (e.g., activity, settings, configuration, users)	✓	✗
	Prioritize based on use of vulnerable method	✓	✓
	Prioritize based on production exploitability	✓	✗
	Correlate risks and threats across phases of the SDLC	✓	✗
	Correlate risks and threats across AppSec scanning engines/tools	✓	✗
 Pipeline Security Controls	Identifies pipeline tooling misconfigurations	✓	✗
	Implements pipeline security controls across the entire pipeline (e.g., least privilege, branch protection, etc.)	✓	✗
	Identifies removal or disabling of pipeline security controls (e.g., SCA scanning)	✓	✗