Software Composition Analysis (SCA) Checklist

6 Key Criteria for Developer-Friendly SCA Solutions

Open source software (OSS) is a critical component of cloud-native application development, allowing developers to get a head start without reinventing the wheel. But OSS—which includes packages, package managers, and package registries—is also a breeding ground for risk. Vulnerabilities are all too common in OSS, as evidenced by recent headline-making vulnerabilities such as Log4j.

To help manage these risks, you can implement open source vulnerability and license compliance scanning with software composition analysis (SCA), but not all SCA providers are created equal.

In this checklist, we'll explore the six key criteria you should look out for when evaluating an SCA provider. Keeping these criteria in mind will help ensure that you're getting the most comprehensive and actionable open source coverage possible.

1. Deep and Trusted Vulnerability Scanning

The premise of vulnerability scanning is simple—identify and document all open source packages within a codebase and then cross-reference those components against vulnerability databases to flag known vulnerabilities. The success of that process, however, hinges on two factors: the scanner's ability to identify all packages and the completeness of the database it cross-references.

The dependency-based nature of OSS makes it extremely important that your vulnerability scanner can identify and document all open source packages in your codebase because dependent packages, also known as transitive dependencies, are just as likely to contain vulnerabilities as direct dependencies. An inability to extrapolate all dependencies for scanning will result in vulnerabilities going undetected.

The ability to uncover all known vulnerabilities is also dependent on the strength of the sources it's fed. It's important to remember that while vulnerabilities are reported daily, they take time to vet, categorize, and publish. This process can create a latency gap wherein new vulnerabilities are known but not yet published to widely distributed vulnerability databases. When using scanners that rely on established databases such as NVD only, there is a chance that your SCA solution isn't flagging vulnerabilities in your OSS because those vulnerabilities are known but not yet published in the database.

Recommended SCA Solution Criteria

- Vulnerability scanning is built on trusted and up-to-date sources such as NVD.
- The vendor maintains its own research capabilities to provide extra coverage by enriching and closing the vulnerability latency gap.
- SCA findings include limitless scanning, no matter how deep the dependency tree, so all direct and transitive packages are scanned for vulnerabilities.

2. A context-aware and Developer-First Approach

Traditional approaches to open source security and compliance tend to surface risk too late in the development lifecycle when open source packages are already integrated into codebases. This reactive approach results in long lists of vulnerabilities and alert fatigue, and that's just one stream of security feedback.

Most organizations leverage multiple point solutions for their infrastructure as code (IaC) security needs, leading to siloed vulnerabilities findings that lack context within the environment. Without the ability to connect findings, it's harder to understand whether a vulnerability is actually exposed and whether it presents real risk.

Both of those challenges result in more effort required from developers and longer investigation times. Developers must then spend time either fixing vulnerabilities or explaining to security teams why they don't pose risks. Thus, traditional approaches to SCA tend to cause friction between developers and security, which is the exact opposite of what cloud-native teams need.

Recommended SCA Solution Criteria

- SCA findings can be surfaced in developers' integrated development environments (IDE) and version control systems (VCS) so that developers can address issues as early in the application development lifecycle as possible.
- SCA checks are embedded as CI/CD build steps before code is deployed as a last line of defense.
- Infrastructure as code (IaC) and open source package scanning tools are consolidated to better prioritize findings and reduce coverage gaps.

3. Deep and Granular Version Bump Fixes

Identifying vulnerabilities is just the first step in eliminating open source risk. The second step involves quickly fixing vulnerabilities found in both transitive and direct dependencies. To fix a vulnerability, you need to update the package to the patched version. One easy, actionable way to update packages at the source is via automated pull request version bumps.

This sounds like a simple task, but as open source packages get updated over time, they may have changed from when they were originally integrated into your codebase. Updating a package could break your application and cause it to not compile or crash, so it's important to know exactly what has changed before you update anything. The safest way to avoid implementing breaking changes with a version bump is to update to the version with minimal changes, but many SCA solutions do not offer that level of granularity.

Recommended SCA Solution Criteria

- Include automated bump fixes that create pull requests to bump each dependency to secure versions.
- Give developers the granularity they need to select the secure package version with minimal changes, thereby reducing the risk of introducing breaking changes.
- Identify and fix vulnerabilities in transitive dependencies to enable complete, expedient fixes.

4. License Compliance

Each organization has its own acceptable use policies for open source licenses, and leveraging open source without complying with license terms can result in hefty fines. SCA solutions help you avoid unintentional license violations by creating an inventory of all open source package licenses you're using so that you can determine which licenses are acceptable and are within your organization's policies.

But flagging license compliance issues is just half the battle. If your SCA solution only provides visibility and auditing, you'll still have to manually remove non-compliant packages. That requires a lot of work for developers who leveraged that particular OSS for a specific reason. They'll need to either write custom code to fulfill the OSS's functionality or find a suitable replacement. That's why it's critical to surface license violation feedback early, but you can also set up systems to avoid this situation altogether.

With a complete SCA solution, you can prevent packages with non-compliant licenses from being integrated or deployed. This way, you can avoid the messy and time-consuming process of removing and replacing those non-compliant packages.

Recommended SCA Solution Criteria

- Surface open source license violation feedback early in the software development lifecycle.
- Include default policies with opinionated levels of severity for common license types, which enable your team to quickly start enforcing licenses.
- Leverage tools with customizable enforcement rules to block packages with non-compliant licenses from being integrated or deployed.

5. Consolidated Software Bill of Materials (SBOM) Generation

An SBOM is a list of the components that make up software or an application—open source packages, IaC resources, etc. The list also details the known vulnerabilities of these components and other metadata, such as the license details for each package.

SBOMs give you visibility into the inventory and posture of the components that make up your cloud-native applications. SBOMs also identify any tampering based on unexpected changes to the components list. Without an SBOM, it's difficult to maintain compliance with regulatory requirements. You may also lack visibility into your system without an SBOM, which prevents you from fully understanding and managing all sources of risk. SBOMs are only as valuable as the sources they are fed, and having multiple sources for multiple types of code and security findings may require manual consolidation and deduplication.

Recommended SCA Solution Criteria

- Include SBOM generation in all major formats, such as CycloneDX or CSV.
- SBOMs include all IaC resources, open source packages, and container images.
- SBOMs also include known licenses and both vulnerability and misconfiguration findings, enriched with data about their severities.

6. Part of a Cloud-Native Application Protection Platform (CNAPP)

The only way to ensure complete coverage when securing cloud-native applications is to scan for vulnerabilities and misconfigurations at each layer and each step of the development lifecycle. Given the complexity of adopting security at each phase of the development lifecycle, the most efficient way to get complete code-to-cloud coverage is to adopt a single, holistic CNAPP solution.

Adopting separate cloud security tools introduces sprawl, leaves coverage gaps, and makes it challenging to address the fact that vulnerabilities can appear across the entire development lifecycle. Surfacing security findings across different channels also results in noise, further distracting developers from their work. And when teams utilize disparate security tools that address infrastructure and application security separately, it's challenging to prioritize issues based on exposure.

Recommended SCA and CNAPP Solution Criteria

- An SCA solution is a component of a singular CNAPP, which consolidates tool sprawl and minimizes coverage gaps.
- Monitor and prevent vulnerabilities throughout the development lifecycle to give you code-to cloud-coverage.
- Take a context-aware approach to open source security and compliance in order to minimize false positives, prioritize findings, and keep code secure faster.

paloalto NETWORKS

3000 Tannery Way
Santa Clara, CA 95054

Main: +1.408.753.4000
Sales: +1.866.320.4788
Support: +1.866.898.9087

www.paloaltonetworks.com