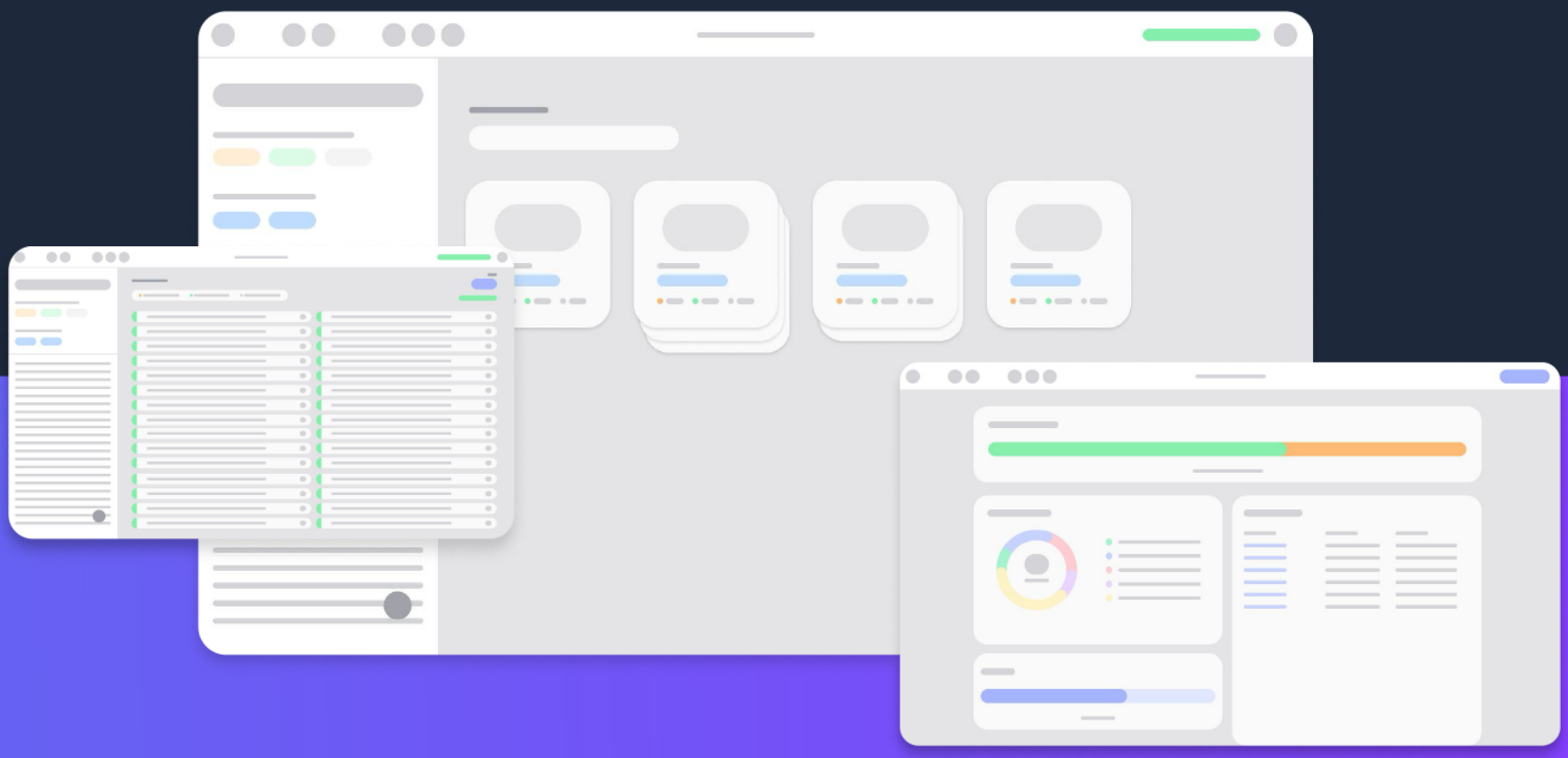
Whitepaper

The Future of Software Supply Chain Transparency

Open source solutions enabling broader SBOM adoption.



SCANOSS

Content

Questions?

Please get in touch through www.scanoss.com

Background

The software supply chain suffers from a persistent lack of transparency. Unlike the hardware industry, which depends on standardised documentation across suppliers, software has developed without a shared infrastructure for identifying components, origins, or licensing. This oversight has introduced significant security, legal, and operational risks.

The LOG4J vulnerability starkly exposed this weakness. The issue was not only the flaw itself, but the widespread inability to determine which systems were affected. This incident highlighted what many already suspected: without visibility into the software we build and use, response efforts are slow, fragmented, and costly.

In 2021, the urgency of this problem prompted the U.S. government to act. Executive Order 14028 mandated the use of Software Bills of Materials (SBOMs) to improve cybersecurity and transparency. Though issued in a national context, the Order set a global precedent, reinforcing the need for standardised approaches to software component tracking.

SBOMs now form the backbone of modern software governance. Standards such as SPDX and CycloneDX are well established, but many organisations still struggle to produce complete and accurate SBOMs—particularly when it comes to the open source components that make up the majority of modern software. Despite growing awareness, gaps remain in detection, coverage, and consistency, making it difficult to gain a full picture of the software in use.

Software Supply Chain Governance

Modern software is rarely built from scratch. It is assembled from external components, often open source, maintained by diverse contributors across the globe. These dependencies introduce hidden risks that can affect the integrity of the final product. Open source components generally fall into two categories: **declared and undeclared.**

Declared open source includes components that are explicitly referenced in metadata or dependency files, making them easier to identify and track. **Undeclared open source**, by contrast, consists of copied or reused code fragments, sometimes modified or stripped of licensing information, that are not formally referenced. This type of code can enter a project unintentionally and go undetected, yet still carries legal and security implications. If not properly identified, undeclared code can compromise compliance, introduce unmitigated vulnerabilities, and create uncertainty around the software's provenance.

To manage this complexity, organisations require reliable inventories of the components in their software. This is what SBOMs provide: structured data about what software is built from, including versions, licences, and origins. A well-integrated SBOM approach enables organisations to monitor compliance, trace component lineage, and respond rapidly to emerging risks.

Effective governance depends on more than simply generating an SBOM. It requires a standardised, automated, and auditable process for producing and validating these inventories across all contributors in the supply chain.

Key Requirements for Effective Governance

Despite growing consensus around the importance of SBOMs, adoption remains uneven. The following capabilities are essential for a scalable, inclusive SBOM ecosystem:

1. Standardisation and Automation

Standardisation and automation are foundational to effective software supply chain governance. Without consistent SBOM generation practices, it is impossible to enforce policies, streamline risk evaluations, or conduct meaningful audits. Governance begins by embedding standard formats, such as SPDX and CycloneDX, and repeatable processes into the development lifecycle, ensuring that every software package, regardless of its source, can be evaluated against a common framework.

2. Traceability and Integrity

SBOMs must not only describe the composition of a software package but also be reliably linked to specific builds and versions. It should be possible to track an SBOM through the software's lifecycle, verifying that it has not been altered or decoupled from its associated code. This traceability can be achieved through digital signatures or decentralised technologies that provide audit trails, reinforcing trust in the provenance of software artefacts.

3. Comprehensive Detection

A complete SBOM must account for both declared and undeclared components. Developers may unintentionally introduce copied or modified open source code, without declaring it or preserving licence information. This can result in significant legal and compliance risks. Governance frameworks must therefore support detection of all software origins, including reused or altered fragments that may otherwise go unnoticed.

4. Integration into Development Workflows

To scale effectively, governance must integrate with modern software development practices. Embedding compliance checks, SBOM generation, and validation into continuous integration and deployment (CI/CD) pipelines ensures that governance becomes automatic and repeatable. This reduces manual overhead while ensuring policies are enforced consistently, without disrupting delivery speed or agility.

The Future of SBOM Tooling

Despite growing awareness of SBOM standards, organisations continue to face challenges in producing complete and verifiable software inventories—particularly when it comes to detecting undeclared open source components. Most tooling still focuses on declared dependencies, leaving out copied, reused, or modified code that lacks formal attribution.

To address this gap, advanced Software Composition Analysis (SCA) must operate at the snippet level, enabling identification of even partial or altered source fragments. This requires a comprehensive and continually updated open source knowledge base. SCANOSS addresses this need by maintaining the largest snippet-level open source database publicly available, enabling accurate detection of both declared and undeclared code across a wide range of software ecosystems.

In practice, this allows development teams and compliance functions to generate more complete SBOMs, critical for legal due diligence, risk assessments, security reviews and quantum readiness. Whether analysing third-party code or internally developed components, the ability to detect partial reuse or licensing gaps ensures that no component goes unnoticed.
To accommodate diverse environments and maturity levels, SCANOSS provides multiple access options:

CI/CD Integration

For organisations with mature DevSecOps practices, SCANOSS integrates directly into **CI/CD pipelines** to generate SBOMs as part of automated builds.

Use case: A DevSecOps team wants to ensure that every software release includes a traceable SBOM, generated at build time.

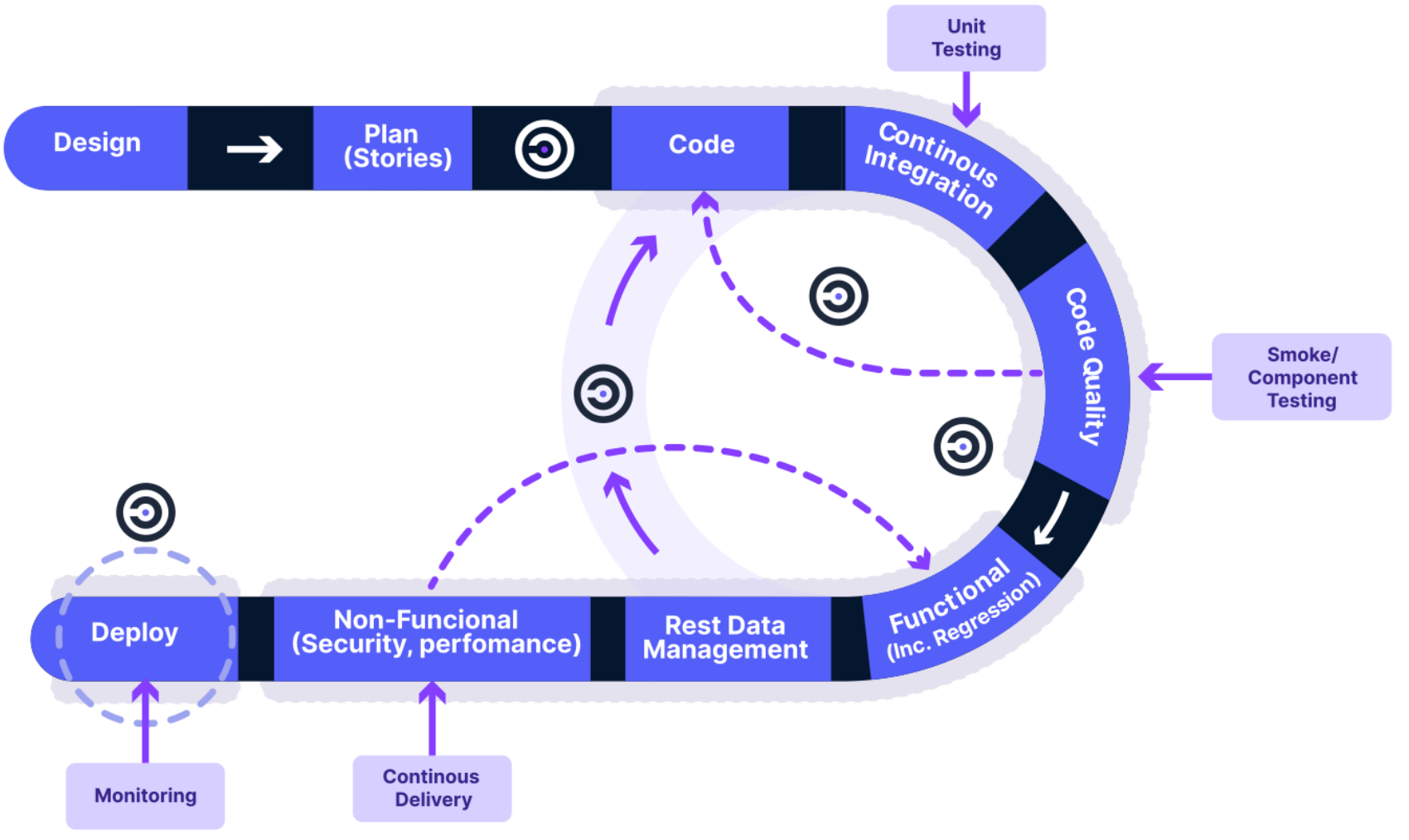
How it works: The SCANOSS engine is triggered during builds in systems like GitHub Actions, GitLab CI, Jenkins, or Azure DevOps. The output can be stored, validated, or published automatically.

Command-Line Interface (CLI)

The SCANOSS CLI provides developers and security teams with direct, scriptable access to SBOM generation from the terminal.

Use case: A developer wants to check for open source issues before committing code, or a security engineer needs to audit a codebase manually.

How it works: You install a CLI tool (like scanoss-py) and run commands locally, such as scanoss scan to generate an SBOM or check for licensing issues.



SCANOSS Workbench

The **SCANOSS Workbench** is a cross-platform desktop application designed for teams needing an accessible, preconfigured tool to scan source code and generate SBOMs locally.

Use case: A procurement or legal team asks a supplier to submit an SBOM with their deliverable. Instead of performing a manual review, the supplier can generate a compliant, standardised SBOM using a prepackaged Workbench.

How it works: The tool can be rebranded and preconfigured with specific knowledge bases or API keys. Suppliers or internal teams simply run it on their computer, no infrastructure or scripting required.

Regardless of how an SBOM is generated, traceability and integrity are critical. SBOMs must remain linked to specific builds and verifiable throughout the software lifecycle.

Together, these capabilities ensure that high-quality SBOMs can be generated at any stage of development or procurement, supporting compliance, transparency, and trust throughout the supply chain.

The Answer is Broad SBOM Adoption

Having SBOM standards in place, like SPDX and CycloneDX, and even an International Standard for Open Source License Compliance (ISO/IEC 5230) by the OpenChain, what remains is to stimulate broad SBOM adoption.

SCANOSS brings an end to decades of costly SCA tools which imposed vendor lock-in mechanisms to software composition data exchange. SMEs and independent developers, a key part of the Software Supply Chain, now have access to leading edge tooling which facilitates software composition, data analysis and exchange. This lowers risks and expenses and finally brings absolute transparency to the Software Supply Chain.

Get involved

The SCANOSS Platform is made entirely available as open source. The collaboration guidelines are available in the source code tree. Questions and suggestions are welcome at https://www.scanoss.com

Get in touch

SCANOSS offers commercial agreements with access to its complete Knowledge Base, additional features and Service Level Agreements.

Please contact info@scanoss.com for further information.

Published on 2025-06-25 by SCANOSS.com

SCANOSS