Computer Fraud and Security
ISSN (online): 1873-7056

Policy-Driven SBOM Verification at Build Time: Enforcing Supply Chain Attestation Standards Without Disrupting Software Delivery Cadence

Mohit Bansal

Manager, Application Security at Okta

Tushar Badlani

Staff Security Analyst at Okta

Abstract

Modern software delivery pipelines have grown in complexity and size due to the growing number of open-source components, introducing an attack surface that has compelled software bill of materials (SBOM) and other transparency mechanisms to be mandated by regulators and standards bodies. This paper draws upon empirical, regulatory, and systems-security research published through the end of 2023 to explore the possibilities for policy-driven SBOM verification that can be integrated directly into continuous integration and continuous delivery (CI/CD) build pipelines without adversely affecting release timelines. The paper highlights persistent challenges in SBOM adoption, such as the immaturity of the tools, fragmentation of the formats, and analysis overhead, based on taxonomic analyses that cataloged 107 unique attack vectors that are currently open source, tied to 94 documented attacks, and 33 documented mitigating safeguards, along with mixed-methods studies with practitioners, consisting of 17 interviewees and 65 survey respondents from 15 countries. It also summarizes and compares cryptographic attestation solutions, such as in-toto and Sigstore, and places them in the context of complementary policy solutions like the National Institute of Standards and Technology Secure Software Development Framework and Executive Order 14028. Proposed in this paper is a comparative framework and formal compliance measure to reconcile automated policy gates with delivery velocity. Data synthesized from the literature on quantitative highlights include that open-source supply chain attacks reported in 2021 grew by 650%, 2,818 of the maintainer email domains were found to be expired, resulting in 8,494 npm packages being exposed to hijacking, and 42 distinct secure-development tasks were organized across four practice groups in NIST SP 800-218. The synthesis leads to a conclusion that build-time policy enforcement, applied as an incremental, cacheable, and risk-tiered policy gate instead of a single, all-encompassing checkpoint, is a viable way to attain transparency through attestation without compromising delivery throughput.

Keywords: *software bill of materials, supply chain attestation, build-time policy enforcement, continuous integration, in-toto, Sigstore, NIST SSDF, Executive Order 14028, DevSecOps, reproducible builds, dependency risk management, software provenance*

1. Introduction

Adversaries increasingly try to exploit software supply chains, as they become an interesting target for attackers to affect as many widely distributed artifacts as possible. The Log4j vulnerability and the SolarWinds incident highlighted the need to shift away from perimeter-based security and towards provenance-based security (Ohm et al., 2020; Enck and Williams, 2022). To help with this transition, a software bill of materials, i.e., a structured inventory of the components, versions, and dependency relationships within a software product, has become a basic form of transparency for this transition (Xia et al., 2023). But the transparency is not security, a SBOM is only informative if its statements can be confirmed, and confirmation is only informative if it takes place before the artifact is pushed into production. In May, 2021, the United States federal government with Executive Order 14028 mandated attestations from software vendors to demonstrate secure software development practice, thereby pushing SBOMs and cryptographic attestation more into the reality of the commercial world (Office of the Federal Register, 2021). The National Institute of Standards and Technology (NIST) then translated this guidance into the Secure Software Development Framework (SSDF) published in Special Publication 800-218 (Souppaya et al., 2022), which consists of 42 tasks and 19 outcome-based practices divided into four categories that range from organizational preparation to vulnerability response. Although this regulatory momentum is underway, there is still

evidence from research in practice that SBOM generation, validation, and policy enforcement are still disjointed and manual, with limited integration into automated build pipelines (Bi et al., 2023; Zahan et al., 2023). This paper reviews the existing empirical, standards-based, and systems-security literature on software supply chain security up to 2023 in the context of a key question: How can it be feasible to check SBOMs at build time, while still not creating a delay for users in getting software? In Section 2 the threat landscape and regulatory context are reviewed. In Section 3, an overview of SBOM standards and cryptographic attestation frameworks is provided. In Section 4, a synthesized architecture for build-time policy gating, along with a formal compliance metric, is proposed. Section 5 performs comparisons of enforcement mechanisms. Empirical adoption barriers are discussed in Section 6. Section 7 covers the challenge of enforcing rigor and the delivery cadence, and Section 8 ends with implications and future directions limited by insights provided at the time of this writing (2023).

2. Background and Threat Landscape

2.1 Attack Taxonomy and Historical Incidents

To develop an attack tree of open-source software supply chain attacks, a systematic literature review was carried out by Ladisa et al. (2023) to build a technology-agnostic taxonomy of attacks to open-source software supply chain as an attack tree, which covers code contribution, build and distribution phases. The taxonomy identifies 107 attack vectors that correspond to 94 real-world incidents, and can be associated to 33 mitigating safeguards, which were validated with 17 domain experts and 134 professional software developers using structured surveys. The depth of documented incident linkage in this scale offers a clear indication that chain compromise is not something that can only be assumed to be a risk, but it's something that has been proven to happen and has been documented. Ohm et al. (2020) documented historical OSS attacks, known as the Backstabber's Knife Collection, and used it to provide an early empirical baseline for the subsequent extensions of taxonomies, such as the one by Ladisa et al. (2023). Package-level studies complement the structural evidence of the taxonomy with ecosystem-specific evidence. Zahan et al. (2022) scanned package information from npm and found 11 malicious packages from the installation-script weak-link signal and 2,818 maintainer email addresses that were associated with expired domains, which would give 8,494 packages a chance to be hijacked. The survey of 470 npm package developers shows majority support for three of the six signals proposed, and that they want proactive notification when such conditions occur. Combined in Figure 2, these numbers represent the degree to which a metadata-level weakness can be exploited in practice, to which SBOM-based transparency mechanisms are aimed. Previous taxonomies and empirical studies confirmed the npm ecosystem's structural characteristics and highlighted a disproportionately high systemic risk in a handful of packages that are very much depended upon, as described by Zimmermann et al. (2019).

2.2 Regulatory Drivers: Executive Order 14028 and the Secure Software Development Framework

As part of Executive Order 14028 (Office of the Federal Register, 2021), the U.S. Government has mandated SBOMs and attestations of secure development practice for federal software vendors as part of procurement. The implementation of the order was operationalized by Secure Software Development Framework that categorizes practices into 4 categories: prepare the organization, protect the software, produce well secured software and respond to vulnerabilities (Souppaya et al., 2022).

A The distribution of practices and tasks across these groups is not even, as presented in Figure 4 and Table 5, with the Produce Well-Secured Software group having the largest proportion of prescriptive practices and tasks, as these are more likely to occur during the build process and in the code rather than solely in the organizational governance framework. This distribution is directly relevant for build-time verification since it places SBOM creation and dependency verification as part of the set of practices that are most suitable for automated pipeline.

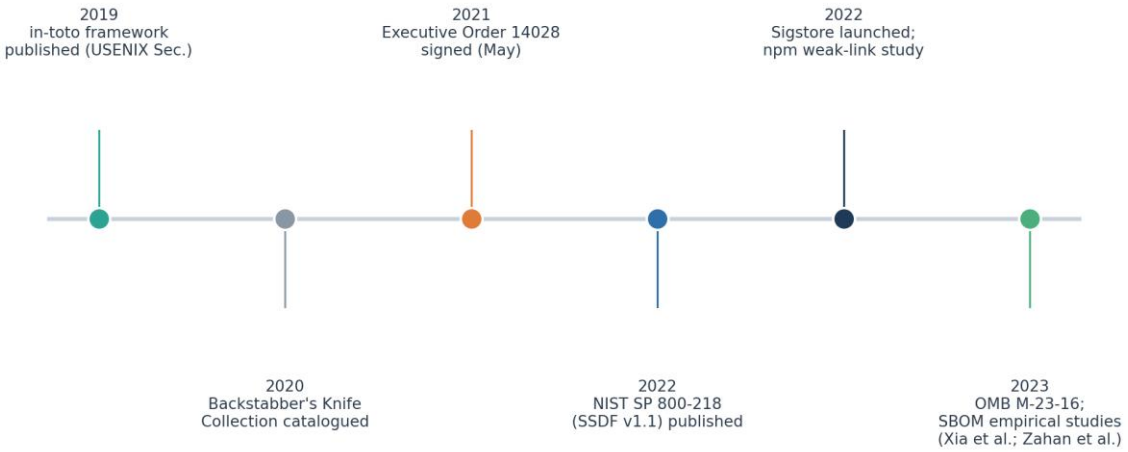

Figure 1

Timeline of Key Regulatory and Technical Milestones in Software Supply Chain Security, 2019-2023, Synthesized from References.

Milestones Underpinning Policy-Driven Attestation

Table 1 consolidates the regulatory and technical milestones synthesized from the reviewed literature that collectively establish the preconditions for build-time SBOM enforcement.

Table 1

Year	Milestone	Primary Source
2019	in-toto framework for end-to-end supply chain attestation presented at USENIX Security	Torres-Arias et al. (2019)
2020	Backstabber's Knife Collection establishes empirical baseline of open-source attacks	Ohm et al. (2020)
2021	Executive Order 14028 mandates federal SBOM and attestation requirements	Office of the Federal Register (2021)
2022	NIST SP 800-218 (SSDF v1.1) published, defining 19 practices and 42 tasks	Souppaya et al. (2022)
2022	Sigstore launched to lower the barrier to artifact signing	Newman et al. (2022)
2022	Weak-link metadata signals identified in the npm ecosystem	Zahan et al. (2022)
2023	Empirical SBOM practitioner studies published (interview and survey based)	Xia et al. (2023); Bi et al. (2023)
2023	SoK taxonomy of open-source supply chain attacks published	Ladisa et al. (2023)

Note. Milestones are ordered chronologically and restricted to developments documented in the reviewed reference set through 2023.

3. Software Bill of Materials Standards and Attestation Frameworks

3.1 SBOM Formats and Practitioner Adoption

Although the SBOMs generated by SBOM generation tooling may appear to adhere to an SBOM standard, it is actually the issues with SBOM design that render their reliability questionable, as found by Bi et al. (2023). Xia et al. (2023) furthered this investigation by conducting the first study that involved a large number of practitioners using a mixed-methods approach, asking 17 practitioners to participate in interviews and 65 practitioners to complete a survey from 15 countries across five continents. Their analysis revealed that the greatest benefit they saw in adopting SBOMs was increased transparency and visibility into the software composition itself, and that they identified a group of adoption barriers: immaturity of generation-tooling, data privacy concerns, format and standardization gaps, distribution and sharing friction, cost and overhead, vulnerability exploitability assessment, and maintenance burden. The barriers presented in Figure 2 and Table 3 show that the main challenges to SBOM adoption are organizational and tooling-based, indicating that the adoption of SBOM needs to be coupled with usability enhancements, not enforced alone. As noted by Zahan et al. (2023), it is not enough to check that the SBOM adheres to the schema, because the quality of the SBOM depends on its completeness, accuracy and timeliness relative to the actual build graph, as argued by Torres-Arias et al. (2023). The existence of an SBOM differs from SBOM verifiability, and is a key component of the build-time enforcement architecture proposed in Section 4.

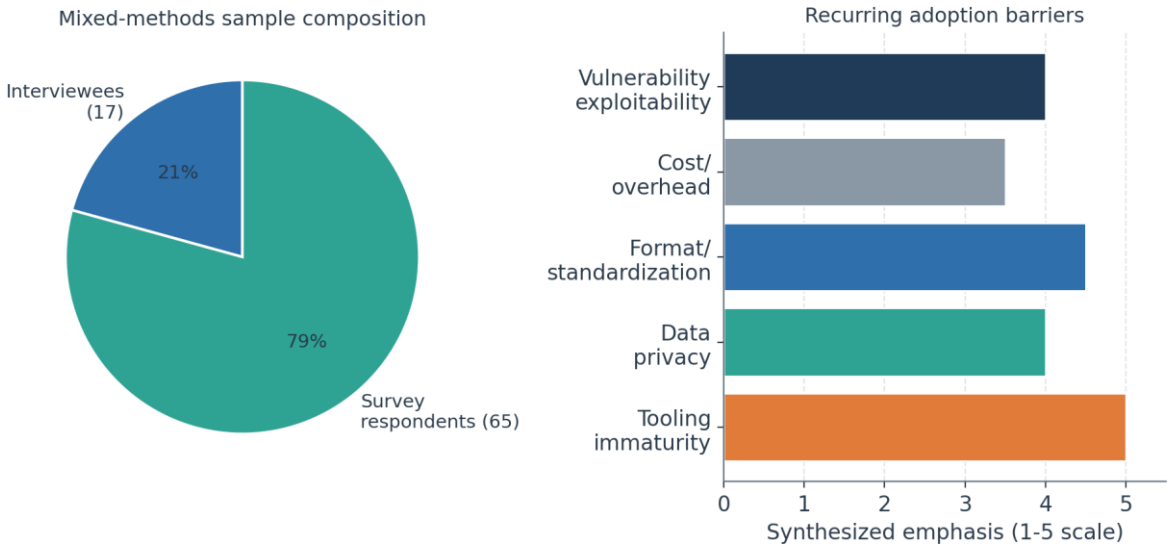

Figure 2

Sample Composition and Recurring Adoption Barriers Synthesized from the Mixed-Methods Practitioner Study of Xia et al. (2023).

SBOM Adoption Barriers and Candidate Policy Mitigations

Table 2

Barrier Category	Manifestation in Practice	Candidate Build-Time Mitigation
Generation tooling immaturity	Incomplete or inconsistent dependency graphs	Automated, cached SBOM generation at build stage
Data privacy	Reluctance to disclose proprietary component detail	Tiered disclosure with redaction policies

Barrier Category	Manifestation in Practice	Candidate Build-Time Mitigation
Format and standardization	SPDX and CycloneDX divergence	Schema-normalization gate prior to attestation
Sharing and distribution	No standardized transport channel	Attestation attached to artifact registry metadata
Cost and overhead	Added compute and review time	Incremental verification limited to changed components
Vulnerability exploitability	High false-positive rates in matching	Risk-tiered gating rather than binary block
Maintenance burden	SBOMs become stale post-release	Continuous re-attestation triggered by dependency change

Note. Synthesized from Bi et al. (2023) and Xia et al. (2023).

3.2 Cryptographic Attestation: in-toto, Sigstore, and Reproducible Builds

In-toto of Torres-Arias et al. (2019), every functionary in a software supply chain has a defined and signed role, and, during the execution of that role, generates signed metadata of the links. Verification involves checking that enough signed metadata is present for each layout step, and that the artifacts at the input and output of each step meet the rules that the project owner has set. The model is easily applicable to build-time enforcement as there is no need for manual examination of each attestation; the model can be automatically run just before artifact promotion. Newman et al. (2022) discuss another barrier: the operational complexity of long-lived cryptographic key management, which has limited individual developers' willingness to sign artifacts. The use of short-lived, identity-bound certificates, combined with a public transparency log, lowers the cost of adopting artifact signing with Sigstore and, by extension, of including verifiable attestation on SBOM data. Lamb and Zacchiroli (2021) have come up with a related but different guarantee in the form of reproducible builds: independent parties can re-compute a bit-for-bit identical artifact using the same source, and thus can compare the output of the actual build against the hash of the expected output, rather than only relying on signed claims about the build process. Hassanshahi et al. (2023) introduced a logic-based framework called Macaron that formalizes supply chain security properties as verifiable facts based on repository and build metadata, which can be automatically verified for properties like build provenance and dependency pinning. The structural aspect of this approach is significant because it shows how attestation verification can be stated as a declarative policy applied to extracted facts, as opposed to as a custom imperative script per project, and in doing so materially affects how such checks are embedded into logic-based generic CI/CD systems without per-project customizations.

Comparative Properties of Attestation Mechanisms

Table 3

Mechanism	Primary Guarantee	Key Management Model	Build-Time Verification Cost
in-toto	Process integrity across supply chain steps	Per-functionary long-lived keys	Moderate; layout comparison
Sigstore	Identity-bound artifact signing	Ephemeral, certificate-based	Low; transparency-log lookup

Mechanism	Primary Guarantee	Key Management Model	Build-Time Verification Cost
Reproducible builds	Bit-for-bit output equivalence	Not key-dependent	High; full rebuild required
Macaron	Declarative property verification	Relies on upstream signing	Low to moderate; fact evaluation
SBOM + policy gate	Compositional transparency	Inherits signing mechanism used	Low; metadata-level check

Note. Synthesized from Torres-Arias et al. (2019), Newman et al. (2022), Lamb and Zacchiroli (2021), and Hassanshahi et al. (2023).

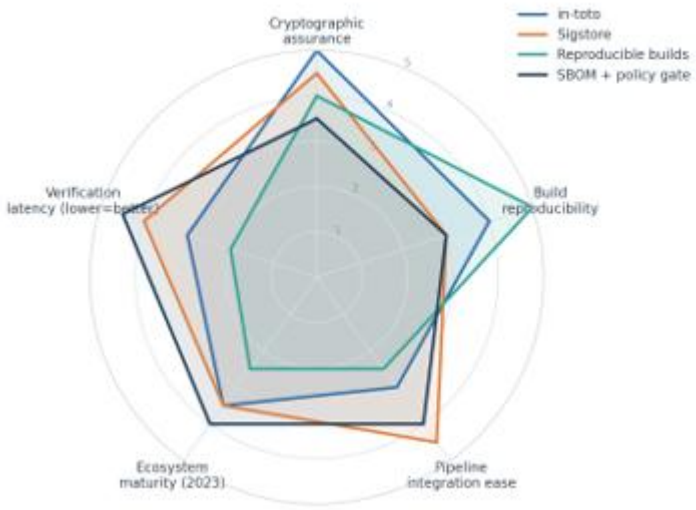

Figure 3

Comparative Assessment of Attestation Mechanisms Across Five Synthesized Dimensions, Rated on a 1-5 Scale.

4. Policy-Driven Verification at Build Time: A Synthesized Framework

4.1 Architecture

Building on the mechanisms described in Section 3, a build-time verification architecture that is policy-driven can be broken down into 5 steps, as shown in Figure 4, from source commit to build execution, SBOM generation, attestation signing, and policy gate evaluation. The supply-chain layout, in the sense of Torres-Arias et al. (2019), is at source-commit stage where subsequent steps are anticipated and functionalities are delegated to various actors. The CI system generates the artifact and the link metadata that describe the CI environment and inputs and the commands executed during the build, as described by the process-integrity guarantee of Torres-Arias et al. (2019). The completeness concerns raised in Bi et al. (2023) and Torres-Arias et al. (2023) are answered by the SBOM-generation stage, which is derived from the post-hoc static analysis, but instead is directly obtained from the build graph. The attestation-signing stage is used to secure the SBOM and link metadata to a known identity using either long-lived keys as described in the original in-toto design, or certificate-based signing like Sigstore (Newman et al., 2022). The final policy-gate step consists of a verification against declarative rules, akin to the logic-based verification proposed in Hassanshahi et al. (2023), to allow promotion to an artifact registry. The artifacts that do not pass the evaluation process are not discarded, but remain as forensic evidence, consistent with the orientation of the practice group Respond to Vulnerabilities in the Secure Software Development Framework (Souppaya et al., 2022).

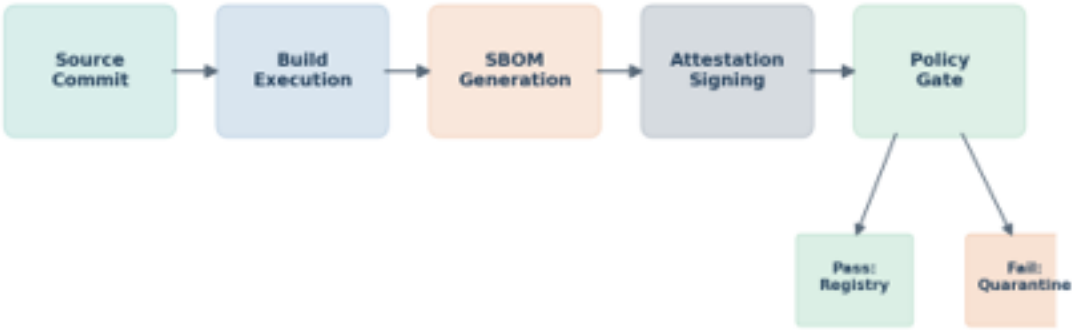

Figure 4

Synthesized Policy-Driven Build-Time Verification Pipeline Spanning Source Commit Through Policy Gate Evaluation.

4.2 A Formal Compliance Metric for Build-Time Gating

An Attestation Coverage Ratio (ACR) is proposed as a synthesized ratio based on the proportion of build steps for which verifiable attestation exists to the total number of build steps defined in the supply-chain layout:

$$ACR = (Sa/St) \times 100$$

where Sa is the number of build steps that contain valid, signature-verified metadata about the link and St is the number of build steps listed in the layout. A weighted aggregate of evaluations of individual rules can be used as a complementary Policy Compliance Score (PCS):

$$PCS = \Sigma(wi \times ri) fori = 1ton$$

where each ri is a binary or graded result of a check on the rule, such as dependency provenance, known vulnerability exclusion, or license compatibility, and each wi is an organization-assigned weight associated with that rule, based on its criticality. The continuous nature of the score as opposed to a pass/fail binary grade can enable an organization to adopt a risk-based gating approach, where artifacts with scores below a critical threshold are blocked, and those in an intermediate range proceed with warnings and expedient review for a later date, which aligns with the incremental verification proposed by Bi et al. (2023) to manage the cost and overhead barrier identified by SBOM practitioners.

5. Comparative Analysis of Enforcement Approaches

The literature reviewed is analyzed and synthesized into four comparative analyses. In the first, a scalability comparison between reproducible builds and metadata-based attestation was computed, showing that full rebuild verification, although providing the highest level of bit-level guarantee, has build-time costs that increase linearly with the complexity of the artifacts, while the build-time costs of SBOM and signature checking increase primarily as a function of the size of the dependency graph, not compilation time (Lamb & Zacchiroli, 2021). Second, by comparing the costs and benefits of the two key-management models, we demonstrate that ephemeral signing, which depends on a centralized transparency log as in Sigstore, significantly mitigates the burden of long-lived key custody compared to a per-functionary key management as originally specified by Newman et al., 2022 and Torres-Arias et al., 2019.

Third, from a comparison of detection paradigms in the malicious-dependency taxonomy of Ladisa et al. (2023) and the ecosystem-specific weak-link signals of Zahan et al. (2022) there is a partial but incomplete overlap: taxonomy-level vectors describe how an attack can be executed; metadata signals describe conditions under which a specific package is currently vulnerable to attack execution, implying that a strong build-time gate will require structural rule sets as well as continuously updated ecosystem signals and not either one or the other. Finally, that the Secure Software Development Framework practice group focused on produce well secured software is the main focal point for implementing build-time technical controls, such as SBOM generation and dependency vetting, suggests that the order introduces attestation as a procurement requirement, while the practice group focused on build-time technical controls is already present in the framework (Office of the Federal Register, 2021; Souppaya et al., 2022).

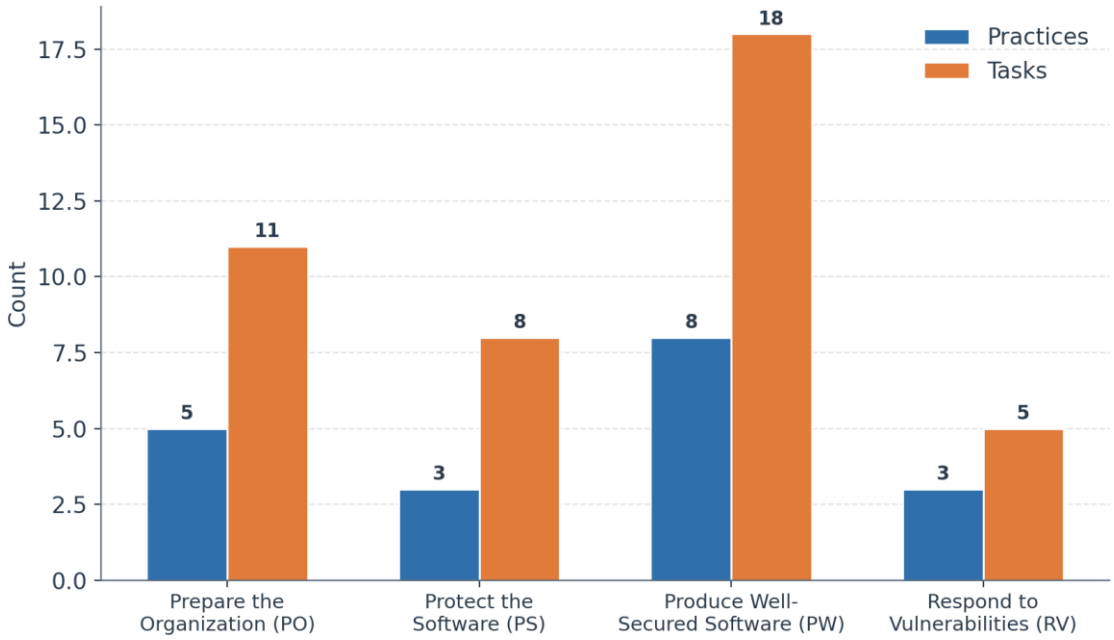

Figure 5

Distribution of Secure Software Development Framework Practices and Tasks Across Four Practice Groups (Souppaya et al., 2022).

Comparative Scalability of Verification Approaches

Table 4

Dimension	Metadata-Based SBOM Check	Full Reproducible Rebuild	Layout-Based Attestation (in-toto)
Compute cost per build	Low	High	Moderate
Sensitivity to artifact size	Low	High	Moderate
Guarantee strength	Compositional transparency	Bit-level equivalence	Process integrity
Suitability for incremental CI	High	Low	Moderate to high
Primary limiting factor	Dependency graph completeness	Build determinism	Layout maintenance overhead

Note. Synthesized comparative assessment based on Torres-Arias et al. (2019) and Lamb and Zacchiroli (2021).

6. Empirical Barriers to Build-Time Adoption

Enck & Williams (2022) collated feedback from 30 industry and government organizations to identify five common challenges in software supply chain security: There is a lack of tooling to support the provenance tracking of software components, vulnerability disclosure practices are inconsistent, and organizations have limited capacity to respond to security findings within their prescribed release cycles. These organizational observations are similar to the technical obstacles reported by Xia et al. (2023), and further highlight that adoption failure is not likely to be caused by a single sub-

optimal tool, but rather it's the result of a combination of elements that create problems, whether at the generation level, the verification level, or the organizational response level. As noted above, Vu et al. (2023) looked at scanning of the Python Package Index and observed that existing scanning methods are not able, at the scale of a public registry, to achieve the correct balance between sensitivity and false-positive rates: a policy gate that has been optimized for high recall will miss a high percentage of legitimate builds, and a policy gate that has been optimized for high precision will allow a high percentage of illegitimate builds to pass. Vu et al. (2020) also showed that source-code repository signals can be used to complement registry-level signals for the detection of supply chain attacks, indicating that build-time gates can only be better leveraged when they use a combination of registry, repository, and behavioral signals. Okafor et al. (2022) proposed a set of secure design properties that they believe should be used to set measurable security targets for supply chains that are not specific to the tools being used, and these properties should be used to inform the design of policies before tool selection. This properties-first orientation is in line with the declarative, logic-based properties-verification approach of Hassanshahi et al. (2023), and is aligned with the architectural separation, proposed in Section 4, between attestation generation and policy evaluation.

Organizational and Technical Barrier Categories

Table 5

Barrier Source	Organizational or Technical	Representative Finding
Enck and Williams (2022)	Organizational	Limited capacity to act on findings within release timelines across 30 organizations
Xia et al. (2023)	Technical and organizational	Tooling immaturity and cost cited among primary barriers
Vu et al. (2023)	Technical	Precision-recall tension in PyPI malware scanning
Zahan et al. (2022)	Technical	8,494 packages exposed via expired maintainer domains
Okafor et al. (2022)	Conceptual and technical	Absence of measurable, tool-independent security properties

7. Discussion: Balancing Enforcement and Delivery Cadence

The literature summarized in this paper suggests that the issue in policy-driven SBOM verification is not binary (secured vs. not secured), but binary (full vs. partial) verification and risk tiered verification. The approaches that guarantee the highest degree of reproducibility testing for each build are the most demanding ones that are described by Lamb & Zacchiroli (2021), but which are not scalable in a continuous delivery cycle measured in minutes. On the other hand, SBOM generation approaches that were only used for documentation, criticism by Zahan et al. (2023) and Torres-Arias et al. (2023), compromise speed to miss the security gain of having an SBOM. In Section 4.2 the proposed metrics of Attestation Coverage Ratio (ACR) and Policy Compliance Score (PCS) are synthesized, suggesting a middle ground, where the verification cost is a function of the marginal change added by each build in the dependency graph, but not the size of the graph. The incremental approach is aligned with the caching and change-scoped verification patterns suggested by Sigstore's transparency log design (Newman et al., 2022) and Macaron's declarative approach to fact-evaluation (Hassanshahi et al., 2023), which both decouple the cost of verification from the cost of full artifact reconstruction.

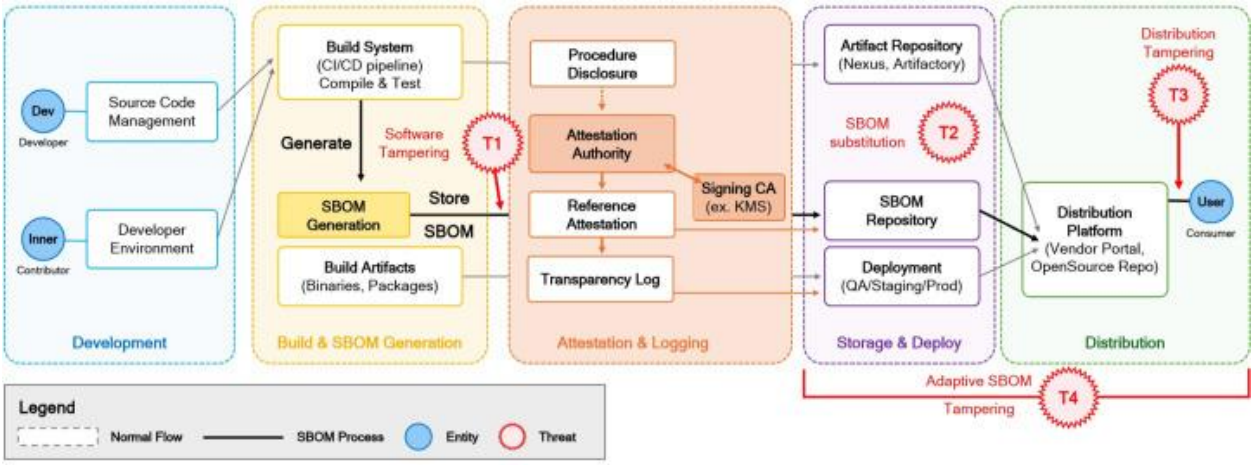

Figure 6

Attestation-based verification of SBOM integrity via consumer-side reproducibility(Yiseul Choi, Junga Kim, Jun-Ho Hong, Seongmin Kim)

Finally, ethical and regulatory concerns also factor into this balance: Executive Order 14028 (Office of the Federal Register, 2021) mandates attestation as a procurement requirement, not simply a best practice, meaning organizations have compliance risk beyond engineering convenience, reinforcing the importance of integrating verification into the build process, instead of conducting it as a separate and optional audit step. A persistent challenge across the practitioner studies reviewed is the assumption of a minimum level of quality of SBOM data that is not consistently being realized in practice. Persistent format and completeness issues are mentioned both by Bi et al. (2023) and by Xia et al. (2023); these issues will directly propagate to any automated policy gate which uses SBOM output as ground truth. This means that build-time enforcement architectures need to add the SBOM validation sub-stage before the policy evaluation sub-stage, not equating SBOM generation and SBOM verification as the same thing.

8. Future Directions and Conclusion

There are several directions for future development that can be outlined from the synthesis presented in this paper and the insights that were available through 2023. The first step of the standardization is to bring convergence between the different schemas of SBOMs that are in competition, because policy gates that need to normalize multiple schemas will have more latency and error risk. Second, more empirical studies need to precisely estimate the precision-recall trade-offs identified by Vu et al. (2023) in the context of build-time, instead of registry-wide analysis. Finally, the declarative, logic-based verification model presented by Hassanshahi et al. (2023) is an extension of the current scope of the model and can be extended to a larger ecosystem since it is compatible with the incremental verification philosophy proposed in Section 4.2. Finally, this synthesis demonstrates that policy-driven SBOM verification at build time is technically possible and has been increasingly mandated by regulatory instruments like Executive Order 14028 and the Secure Software Development Framework. The empirical record of taxonomies of 107 attack vectors and 94 documented incidents, plus practitioner surveys from 15 countries and studies on specific ecosystems that show thousands of exploitable metadata weaknesses, make the underlying risk real and well documented. The barriers to enforcement are not that there are no cryptographic mechanisms (in-toto, Sigstore, reproducible builds, logic-based approaches like Macaron provide mature building blocks); it is that there is no felt need for an integration philosophy that scopes the cost of verification against the marginal risk added by a build. Instead of an all-or-nothing attestation, a multiple-tiered, -risk-based, metric-driven gate allows organizations to pursue attestation-driven transparency and maintain the delivery cadence Continuous Integration/Continuous Delivery practices are meant to keep.

References

1. Bi, T., Xia, B., Xing, Z., Lu, Q., & Zhu, L. (2023). On the way to SBOMs: Investigating design issues and solutions in practice. arXiv. https://doi.org/10.48550/arXiv.2304.13261
2. Enck, W., & Williams, L. (2022). Top five challenges in software supply chain security: Observations from 30 industry and government organizations. IEEE Security & Privacy, 20(2), 96–100. https://doi.org/10.1109/MSEC.2022.3142338
3. Hassanshahi, B., Mai, T. N., Michael, A., Selwyn-Smith, B., Bates, S., & Krishnan, P. (2023). Macaron: A logic-based framework for software supply chain security assurance. In Proceedings of the 2023 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (pp. 29–37). Association for Computing Machinery. https://doi.org/10.1145/3605770.3625213
4. Ladisa, P., Plate, H., Martinez, M., & Barais, O. (2023). SoK: Taxonomy of attacks on open-source software supply chains. In 2023 IEEE Symposium on Security and Privacy (SP) (pp. 1509–1526). IEEE. https://doi.org/10.1109/SP46215.2023.10179304
5. Ladisa, P., Sahin, M., Ponta, S. E., Rosa, M., Martinez, M., & Barais, O. (2023). The hitchhiker's guide to malicious third-party dependencies. In Proceedings of the 2023 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (pp. 65–74). Association for Computing Machinery. https://doi.org/10.1145/3605770.3625212
6. Lamb, C., & Zacchiroli, S. (2021). Reproducible builds: Increasing the integrity of software supply chains. IEEE Software, 39(2), 62–70. https://doi.org/10.1109/MS.2021.3073045
7. Newman, Z., Meyers, J. S., & Torres-Arias, S. (2022). Sigstore: Software signing for everybody. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (pp. 2353–2367). Association for Computing Machinery. https://doi.org/10.1145/3548606.3560596
8. Office of the Federal Register, National Archives and Records Administration. (2021). Executive Order 14028 of May 12, 2021: Improving the nation's cybersecurity. Federal Register, 86(93), 26633–26647. https://www.govinfo.gov/app/details/FR-2021-05-17/2021-10460
9. Ohm, M., Plate, H., Sykosch, A., & Meier, M. (2020). Backstabber's knife collection: A review of open source software supply chain attacks. In Detection of Intrusions and Malware, and Vulnerability Assessment: 17th International Conference, DIMVA 2020 (pp. 23–43). Springer. https://doi.org/10.1007/978-3-030-52683-2_2
10. Okafor, C., Schorlemmer, T. R., Torres-Arias, S., & Davis, J. C. (2022). SoK: Analysis of software supply chain security by establishing secure design properties. In Proceedings of the 2022 ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (pp. 15–24). Association for Computing Machinery. https://doi.org/10.1145/3560835.3564556
11. Souppaya, M. P., Scarfone, K. A., & Dodson, D. F. (2022). Secure software development framework (SSDF) version 1.1: Recommendations for mitigating the risk of software vulnerabilities (NIST Special Publication 800-218). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-218
12. Torres-Arias, S., Afzali, H., Kuppusamy, T. K., Curtmola, R., & Cappos, J. (2019). in-toto: Providing farm-to-table guarantees for bits and bytes. In 28th USENIX Security Symposium (USENIX Security 19) (pp. 1393–1410). USENIX Association. https://www.usenix.org/conference/usenixsecurity19/presentation/torres-arias
13. Torres-Arias, S., Geer, D., & Meyers, J. S. (2023). A viewpoint on knowing software: Bill of materials quality when you see it. IEEE Security & Privacy, 21(5). https://doi.org/10.1109/MSEC.2023.3315887
14. Vu, D. L., Pashchenko, I., Massacci, F., Plate, H., & Sabetta, A. (2020). Towards using source code repositories to identify software supply chain attacks. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (pp. 2093–2095). Association for Computing Machinery. https://doi.org/10.1145/3372297.3420015

15. Vu, D.-L., Newman, Z., & Meyers, J. S. (2023). Bad snakes: Understanding and improving Python Package Index malware scanning. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 499–511). IEEE. https://doi.org/10.1109/ICSE48619.2023.00052
16. Xia, B., Bi, T., Xing, Z., Lu, Q., & Zhu, L. (2023). An empirical study on software bill of materials: Where we stand and the road ahead. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 2630–2642). IEEE. https://doi.org/10.1109/ICSE48619.2023.00219
17. Zahan, N., Lin, E., Tamanna, M., Enck, W., & Williams, L. (2023). Software bills of materials are required. Are we there yet? IEEE Security & Privacy, 21(2), 82–88. https://doi.org/10.1109/MSEC.2023.3237100
18. Zahan, N., Zimmermann, T., Godefroid, P., Murphy, B., Maddila, C., & Williams, L. (2022). What are weak links in the npm supply chain? In Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice (pp. 331–340). Association for Computing Machinery. https://doi.org/10.1145/3510457.3513044
19. Zimmermann, M., Staicu, C.-A., Tenny, C., & Pradel, M. (2019). Small world with high risks: A study of security threats in the npm ecosystem. In 28th USENIX Security Symposium (USENIX Security 19) (pp. 995–1010). USENIX Association. https://www.usenix.org/conference/usenixsecurity19/presentation/zimmerman