A Zero-Knowledge Framework for Confidential and Verifiable SBOM Validation

Jiarou Deng, Yang Yang, Michael Rushanan

Johns Hopkins University | Whiting School of Engineering | Baltimore, MD

Abstract

New regulations require Medical Device Manufacturers (MDM) to publicly share Software Bills of Materials (SBOM) or share them with their users, improving supply chain transparency. An unexpected consequence of this transparency, however, is the exposure of component details that attackers can exploit. We introduce a Zero-Knowledge Proof (ZKP) framework that enables stakeholders to *verify SBOM properties (e.g., "no critical vulnerabilities") without revealing sensitive software components or their associated details*.

Transparency = Attack Surface

- **Problem:** SBOM transparency improves accountability but exposes software composition and dependency details that may serve as avenues for exploiting information.

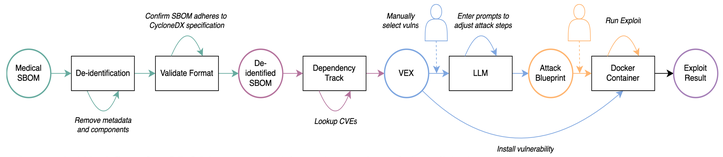

Figure 1: Workflow for SBOM-to-attack-blueprint and exploit validation. [1]

- We implemented a semi-automated LLM pipeline that:
  - De-identified medical device SBOMs
  - Performed CVE lookups using the NVD
  - Created attack blueprints using CVES + LLM
  - Validated blueprints using Docker containers

- We determined that **77.8%** of the CVEs we identified and the attack blueprints we generated were successfully exploited.

CVE	Successful Exploit	Severity	AV	AC	PR	UI	S	C	I	A
CVE-2023-46308	No (22.2%)	Critical (11.1%)	N (77.8%)	L (100%)	N (77.8%)	N (77.8%)	U (88.9%)	H (44.4%)	H (44.4%)	H (44.4%)
CVE-2023-48631	Yes (77.8%)	High (88.9%)	N	L	N	N	U	N (44.4%)	N (55.6%)	L (22.2%)
CVE-2024-4367	No	High	N	L	N	R (22.2%)	U	H	H	H
CVE-2022-25883	Yes	High	N	L	N	N	U	N	N	L
CVE-2023-45133	Yes	High	L (22.2%)	L	L (11.1%)	N	C (11.1%)	H	H	H
CVE-2024-23331	Yes	High	N	L	N	N	U	L (11.1%)	N	N (33.3%)
CVE-2023-34092	Yes	High	N	L	N	N	U	H	N	N
CVE-2023-26115	Yes	High	N	L	N	N	U	N	N	H
CVE-2023-30533	Yes	High	L	L	N	R	U	H	H	H

TABLE 1: CVEs analyzed in the case study with exploit success, severity, and CVSS 3.1 base metrics. Percentages give the share of CVEs with that value and are shown only on the first occurrence per CVSS column (AV, AC, PR, UI, S, C/I/A).

- We conclude that SBOM transparency *creates an attack surface.*

- **Motivation:** find a solution to verify SBOM properties without disclosing sensitive supply chain details.

ZKP for SBOM Validation

Solution: We propose a zero-knowledge proof (ZKP) framework for SBOM validation that ensures transparency without compromising confidentiality. A **prover–verifier protocol** enables SBOM producers to generate proofs of specific security assertions, which can be independently verified by SBOM consumers using only public parameters. In particular:

1. Producers (provers) generate commitments to an SBOM and then publish non-interactive zero-knowledge proofs.
2. Proofs demonstrate that specific security assertions about the SBOM are accurate.
3. Consumers (verifiers) confirm claims using only public parameters without exposing the underlying SBOM or sensitive build data.

Protocol Properties:

1. *Privacy-Preserving.* Provides a cryptographic mechanism to reconcile transparency with confidentiality.
2. *Confidential Assurance.* Validates security assertions without disclosing the SBOM's software components, dependencies, or sensitive metadata.
3. *Verifiable Provenance.* Enables anyone to verify the origin, linkage, and authenticity of claims using only public parameters.

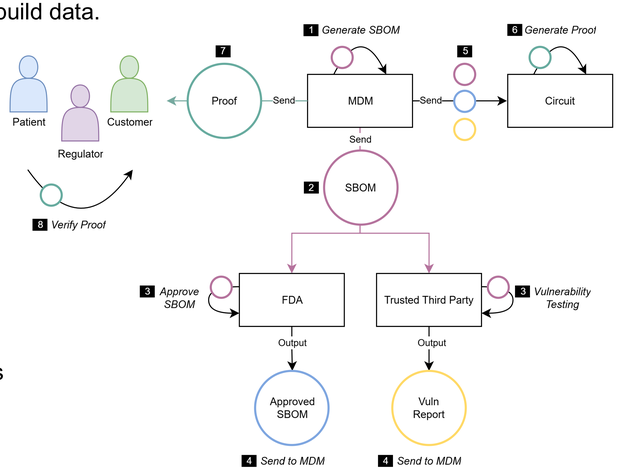

Figure 2: Generalized SBOM-to-proof use case flow.

Protocol Users:

- *MDM.* Generates the SBOM, commits to it cryptographically, and publishes proofs of report validity.
- *Trusted Third Party.* Conducts vulnerability analysis and issues signed assessment reports.
- *Regulator.* Validates proofs and ensures regulatory compliance.
- *Customer or Patient.* Verifies proofs to confirm product security without SBOM disclosure.

ZKP Framework Use Cases:

- *Vulnerability Presence Check.* A regulator, customer, or patient wants to verify whether an SBOM includes any vulnerabilities without accessing the SBOM.
  "This SBOM contains no components with CVES as of time T."

- *Specific Vulnerability Exclusion.* A customer or user needs assurance that a known critical vulnerability (e.g., Log4j) is *not present* in the supplied software.
  "This SBOM does not include components affected by CVE-X"

- *Exploitability Verification.* An MDM wants to show that no known vulnerability is exploitable.
  "All vulnerabilities are unreachable or mitigated by design."

Use Case 2: Prototype Implementation

We implemented cryptographic proofs using the Merkle TreeGap construction to validate claims from third-party vulnerability reports.

Key idea: Each report is encoded as a sorted Merkle tree of component hashes and their vulnerability states.

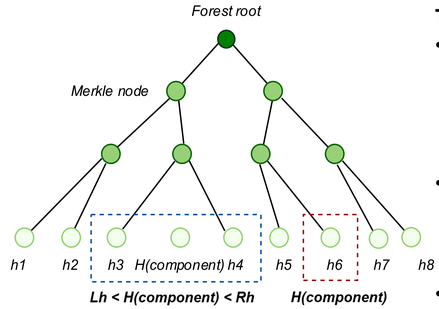

Figure 3: Illustration of Merkle TreeGap approach.

TreeGap proof:

- Shows whether a component lies inside / outside a designated subset (e.g., exploitable vulnerabilities)
- Uses neighboring hashes ("gaps") to prove position without revealing the full report
- Enables membership / non-membership proofs in zero knowledge

Discussion

While our design establishes a three-party workflow (MDM, trusted third-party, and client) with regulator's oversight, several open challenges remain:

- *Data normalization and interoperability.* Inconsistent SBOM naming and version formats; scopes (e.g., <2.1.3) and pre-release versions complicate deterministic matching.

- *Scalability and efficiency.* Proof generation must handle large SBOMs and reports efficiently; we adopt batch proofs, Merkle forests, and Groth16.

- *Snapshot freshness and reissuance.* Vulnerability intelligence evolves rapidly. Proofs need to be bound to specific vulnerability snapshots to support timely reissuance.

- *Governance and trust distribution.* Defining the roles of MDMs, trusted third parties, and regulators remains an open governance question.

Reference: [1] J. Deng, Y. Yang, and M. Rushanan, "SBOM Transparency v. Exposure Dilemma: A Case Study on Adversarial Access to Public SBOMS in Healthcare," In the proceedings of the 2025 Workshop on Cybersecurity in Healthcare (HealthSec '25), 2025. **Acknowledgments**: We thank Dorian Liu <tliu96@jh.edu> for his contributions to this project.