Secure Software Supply Chain with Software Bill of Materials (SBOMs)

Module 7 Lab Report

Student Name: Yosef Stifanos
Date: November 21, 2024
Course: ENG 298 - Cybersecurity Fundamentals

Introduction

This lab applied Security Engineering principles to analyze the software supply chain of the NG911 (Next Generation 9-1-1) application repository. Using GitHub Codespaces and open-source tools (Syft, Trivy, and Grype), I generated Software Bills of Materials (SBOMs) in two different formats and performed vulnerability analysis to identify security risks within the software dependencies. This exercise demonstrates how transparency and component visibility support secure design, verification, and lifecycle assurance in modern software systems.

Part 1: SBOM Generation Results

Tools and Formats Used

Two industry-standard tools were used to generate SBOMs for the NG911 repository:

1. **Syft** - Generated SBOM in SPDX JSON format
2. **Trivy** - Generated SBOM in CycloneDX JSON format

Component Count Comparison

Tool	Format	Components Identified
Syft	SPDX JSON	109
Trivy	CycloneDX JSON	220

Analysis of Differences

Key Observation: Trivy identified more than double the components (220) compared to Syft (109). This significant difference can be attributed to several factors:

1. **Scanning Depth:** Trivy appears to catalog transitive dependencies (dependencies of dependencies) more comprehensively, while Syft may focus on direct dependencies.

2. **Format Differences:** The SPDX format generated by Syft tends to be more focused on license compliance and direct package relationships, while CycloneDX (used by Trivy) is designed for security analysis and includes more granular component details.
3. **Detection Methodology:** Trivy performs filesystem scanning that may detect OS-level packages and libraries that Syft's analysis doesn't capture, particularly for Python applications where virtual environments and system packages can overlap.

SPDX vs. CycloneDX Format Differences

One Key Difference: The SPDX format emphasizes **license compliance** and package relationships with detailed fields for licensing information, copyright holders, and file-level checksums. In contrast, CycloneDX prioritizes **security use cases** with richer vulnerability tracking metadata, component pedigree information, and explicit fields for known vulnerabilities. This makes SPDX better suited for legal/compliance teams, while CycloneDX is optimized for security and DevSecOps workflows.

Part 2: Vulnerability Analysis

Grype Scan Results

The Grype vulnerability scanner analyzed the Syft-generated SBOM and identified **8 vulnerabilities** across the NG911 software dependencies:

- **0 Critical** severity
- **2 High** severity
- **5 Medium** severity
- **1 Low** severity

Top 5 Vulnerabilities

CVE/GHSA ID	Severity	Component	Version	Comment
GHSA-5rjg-fvgr-3xxf	High	setuptools	72.1.0	Path traversal vulnerability allowing arbitrary file writes; fixed in 78.1.1
GHSA-2qfp-q593-8484	High	brotli	1.1.0	Compression library vulnerability requiring update to 1.2.0
GHSA-9hjg-9r4m-mvj7	Medium	requests	2.32.3	HTTP library vulnerability in cookie handling; fixed in 2.32.4
GHSA-pq67-6m6q-mj2v	Medium	urllib3	2.2.2	URL parsing vulnerability in HTTP client library; fixed in 2.5.0

CVE/GHSA ID	Severity	Component	Version	Comment
GHSA-48p4-8xcf-vxj5	Medium	urllib3	2.2.2	Additional vulnerability in same urllib3 version; fixed in 2.5.0

Detailed Analysis: Selected CVE

Selected Vulnerability: GHSA-5rjg-fvgr-3xxf (CVE-2025-47273)

Component: setuptools version 72.1.0

NVD Summary: This High severity vulnerability (CVSS 7.7) involves a path traversal flaw (CWE-22) in setuptools' PackageIndex component that allows attackers to write files to arbitrary filesystem locations with the permissions of the running Python process. The vulnerability can be exploited through malicious URLs on package index pages, potentially escalating to remote code execution when installing packages from untrusted sources.

Remediation: Update setuptools to version 78.1.1 or later immediately.

Risk Context: While the EPSS score indicates only a 0.09% probability of exploitation in the next 30 days (26th percentile), the High severity rating means successful exploitation could compromise the entire development or deployment environment where packages are installed.

Key Observations

1. **Network Library Concentration:** Most vulnerabilities affect HTTP/networking libraries (requests, urllib3, brotli), which are common attack vectors for supply chain compromises.
2. **urllib3 Requires Immediate Attention:** Two separate vulnerabilities (GHSA-pq67-6m6q-mj2v and GHSA-48p4-8xcf-vxj5) affect the same urllib3 version (2.2.2), making this component the highest priority for patching.
3. **Patch Availability:** All identified vulnerabilities have patches available, enabling straightforward remediation through dependency updates.
4. **Low Active Exploitation:** All vulnerabilities show EPSS scores below 1.1%, indicating these are not currently being actively exploited in the wild, providing a window for proactive remediation.

Reflection: Security Engineering Insights

The Value of Transparency

This lab demonstrated how SBOMs provide critical visibility into software composition—a fundamental requirement for secure system design. Before generating these SBOMs, the NG911 application was essentially a "black box" of dependencies. The SBOM transformed it into a transparent, auditable inventory. This visibility directly supports the **open design principle** from Saltzer and Schroeder: security mechanisms should not

depend on secrecy of design but rather on the secrecy of keys or credentials. By openly documenting what components exist, we can verify their integrity and maintain them properly.

Defense in Depth Applied to Supply Chain

The lab illustrated **defense in depth** at the supply chain level. Rather than trusting that upstream packages are secure, we:

1. Generated comprehensive inventories (SBOMs)
2. Cross-validated with two different tools (Syft and Trivy)
3. Performed vulnerability scanning (Grype)
4. Mapped findings to authoritative sources (NVD)

Each layer provides additional assurance. If one tool missed a component or vulnerability, the other might catch it. This redundancy is essential when dealing with complex dependency trees where a single vulnerable component could compromise an entire system.

Least Privilege and Complete Mediation

The urllib3 and setuptools vulnerabilities highlight why **least privilege** matters in dependency management. These libraries have broad filesystem and network access because they're fundamental to Python's package ecosystem. When vulnerabilities exist in such privileged components, attackers can leverage that access for exploitation. The **complete mediation** principle—checking every access to every resource—becomes crucial here. SBOMs enable complete mediation by ensuring we know every component that could request access to resources.

Practical Implications for NG911 Systems

For a critical system like NG911 (emergency services), these vulnerabilities aren't academic concerns. A compromised emergency dispatch system could:

- Delay emergency response
- Expose sensitive caller information
- Disable critical infrastructure during emergencies

The two High severity vulnerabilities (setuptools and brotli) should be patched immediately. The urllib3 vulnerabilities are particularly concerning because HTTP libraries in emergency systems handle sensitive location data, caller information, and inter-agency communications.

Lessons Learned

1. **Tool Variation Matters:** The 101-component difference between Syft and Trivy demonstrates that no single tool provides complete visibility. Using multiple SBOM generators increases coverage.

2. **Continuous Monitoring Required:** SBOMs are point-in-time snapshots. New vulnerabilities are discovered daily (the CVE system processed over 25,000 vulnerabilities in 2023 alone). SBOM generation and vulnerability scanning must be integrated into CI/CD pipelines for continuous assurance.
3. **Actionable Intelligence:** The real value isn't just knowing vulnerabilities exist—it's having specific remediation paths. Knowing that updating setuptools from 72.1.0 to 78.1.1 resolves GHSA-5rjg-fvgr-3xxf provides actionable intelligence for security teams.
4. **Supply Chain as Attack Surface:** Modern applications are composed primarily of third-party code. The NG911 application's 109-220 components (depending on the tool) means the attack surface extends far beyond code written by the development team. SBOMs make this expanded attack surface visible and manageable.

Connection to Course Concepts

This lab reinforced several key Security Engineering concepts from Module 7:

- **Assurance:** SBOMs provide evidence that we know what's in our systems, supporting security assurance activities.
- **Verification:** Vulnerability scanning verifies that known-bad components aren't present (or flags them if they are).
- **Lifecycle Management:** Regular SBOM regeneration supports secure maintenance throughout a system's operational lifetime.
- **Risk Management:** The severity and EPSS scores enable risk-based prioritization of remediation efforts.

Conclusion

This lab transformed abstract supply chain security concepts into concrete, actionable practices. By generating SBOMs for the NG911 repository and analyzing them for vulnerabilities, I gained hands-on experience with tools and processes that are becoming mandatory in many industries (e.g., Executive Order 14028 requires SBOMs for federal software procurement). The identification of 8 vulnerabilities, including 2 High severity issues, demonstrates why software transparency matters—you can't secure what you can't see. For critical infrastructure like emergency services, this visibility isn't optional; it's a prerequisite for trustworthy operation.