SBOMs in the Windows Supply Chain

an SPDX Success Story

Joe Bussell
Windows Engineering System | Tool Benders
jobussel@microsoft.com

♂ My pronouns are he/him (why this matters)

Book a meeting with me
Inviting feedback
Time zone: (GMT-8) US Pacific
Agenda

- Introduction to the Windows supply chain
- Overview of SPDX SBOMs and their role in software supply chain trust
- Understanding Provenance in the context of Windows packages
- Validating SPDX SBOM signatures
- Validating package content hashes against SBOM hashes
- Combining Provenance and signed SBOMs for enhanced observability
- Challenges and potential improvements
- Conclusion and next steps
Introduction to the Windows supply chain

Windows Update, Microsoft Update Catalog, Microsoft Digital Operations Center, etc.
Overview of SPDX SBOMs and their role in software supply chain trust

**SBOM**: Provides transparency in software composition, listing components, dependencies, libraries, and more.

**SPDX**: An industry-standard format for SBOMs, created by the Linux Foundation, ensuring easy sharing and comparison of data.

Key Benefits:

- Transparency: Gain insight into software components for security and compliance assessment.
- Vulnerability Management: Identify and manage known vulnerabilities and risks efficiently.
- License Compliance: Ensure legal requirements are met and prevent potential issues.
- Supply Chain Risk Management: Make informed decisions about the trustworthiness of software components.

**SPDX SBOMs in Windows**: Crucial for managing dependencies, identifying security vulnerabilities, and tracking component provenance in the Windows build process.
Understanding Provenance in the context of Windows packages

**Provenance**: Origin and history of a software component, including development, source repository, build process, and distribution.

**Strong Provenance**: A provenance.json file containing Git commit hash, build ID, and repo ID whose hash is protected by a signed SBOM.

**Importance of Provenance**:
- Authentication: Verify component authenticity and prevent tampering.
- Auditability: Enable traceability for audits and compliance checks.
- Security: Identify and manage potential security risks.
- Decision-making: Make informed decisions based on component trustworthiness.

**Trusted Package Principles**:
- Provenance: Accountable party and/or source repository.
- Tamper Protection: Signed content and SBOM by a trusted root authority.
- Observability/Transparency: SBOM with known 1\textsuperscript{st} and 3\textsuperscript{rd} party dependencies.
Validating SPDX SBOMs

**Objective**
- Ensure SBOMs meet usability, compliance, and security requirements.

**Package Validation**
- Enforces SBOM signature validation using CoseSignTool.
- Uses open-source SBOM library for SPDX 2.2 validation [https://github.com/microsoft/sbom-tool](https://github.com/microsoft/sbom-tool).
- Validates SBOM hashes match file hashes, no extra files, no missing files.
- Enforces Provenance validity; component owner defined.
Validating Evidence Signatures

Objective
• Ensure all evidence is secured with a valid digital signature.

COSE Signatures
• CBOR Object Signing and Encryption (COSE) used for SBOM signing and other supply chain artifacts.
• Concise Binary Object Representation (CBOR) is data format designed for small code size and small message size.

SBOM Signature Validation:
• Contents of the SBOM file match the signature in the detached signature file (.cose).
• SBOM signed by a certificate that chains up to a trusted root CA (e.g., Microsoft Supply Chain RSA Root CA 2022).

CoseSignTool
• Platform-agnostic command-line application to COSE sign files and validate COSE signatures.
• Coming soon in the Windows ADK and WDK.
• Open source for this tool coming soon! [microsoft/CoseSignTool: CoseSignTool is a platform-agnostic command line application to COSE sign files and validate COSE signatures. CoseParser is a library of functions for COSE signing and validation for use by .NET applications.](github.com)

CoseParser
• Library of functions for COSE signing and validation in .NET applications.
• Based on [NuGet Gallery | System.Security.Cryptography.Cose 7.0.0.](github.com)

Detached vs Embedded Signatures
• Detached: Separate from the source file but contains the file's hash for validation.
• Embedded: Contains an encoded copy of the file's content; requires a COSE-aware application for reading and validation.
Validating package content hashes against SBOM hashes

Objective
• Ensure package contents are untampered by comparing content hashes with SBOM hashes.

Hash Calculation
• Generate hashes for each package file using a cryptographic hash function (e.g., SHA-256).

SBOM Hash Extraction
• Extract hashes of package contents from the SBOM.

Hash Comparison
• Compare calculated package content hashes with extracted SBOM hashes.
• Any files missing? Any added files?

Validation Result
• Report discrepancies or successful validation for each package.

Tamper Detection
• Identify and investigate discrepancies to detect potential tampering.
Objectives:

- The Windows software supply chain is observable, from dependencies to updates.

Strong Provenance

- Traceability and accountability of package origin (e.g., Git commit hash, build ID, and repo ID).

Signed SBOMs

- Digitally signed SBOMs ensure integrity and authenticity of package contents.

Enhanced Observability

- Combining provenance and SBOMs for comprehensive understanding of the supply chain.

Increased Trust

- Improved security posture and confidence in the software supply chain.

Efficient Auditing

- Facilitates streamlined risk assessment and mitigation processes.

Collaborative Effort

- Cross-team alignment and shared responsibility in maintaining software supply chain trust.
Challenges and potential improvements

- Address gaps in Provenance and SBOM coverage in open source, 1st, and 3rd party packages.
  - Identify source for all components and tools. No more SOUP.
  - Identify component and tool owners, enhance cross-team collaboration on SDL best practices.
  - SBOMs created for all packages: makecab, NuGet, VCPkg, VSIX, containers, Windows Store apps, etc.

- Validate Provenance and SBOM at every stage of the software supply chain.

- Streamline SBOM creation and validation through open-source tool collaboration.

- Create just-in-time risk validation against Provenance and SBOMs (VEX?).

- Publish the Windows SBOM.

- Packages on the Microsoft Digital Operations Center should have accurate SBOMs that OEM partners can download and verify.
A way to Link Async Repo Scans to Product SBOMs

Async Repo Analysis at Each Stage, undocked, Windows, & Updates

Build at Each Stage, undocked, Windows, & Updates

Validate input SBOMs

Signed SBOM

Registry

Build Output + SBOM + Provenance

Release Artifact

Async Code Scan

Provenance
- Repo ID
- Git Commit Hash

Repository

Claim

Identity

Artifact

Evidence
Conclusion and next steps

• Leverage Provenance and signed SBOMs for enhanced observability.
• Prioritize addressing gaps in Provenance and SBOM coverage.
• Foster cross-team collaboration and shared best practices.
• Streamline SBOM creation and validation through open-source tool collaboration.
• Cache evaluation data, if the SBOM has the same hash we need not recompute.
• Implement just-in-time risk validation for proactive security.
• Monitor progress, refine processes, and adapt to new challenges.
Questions?
Additional Information

Available now:

• **microsoft/sbom-tool**: The SBOM tool is a highly scalable and enterprise ready tool to create SPDX 2.2 compatible SBOMs for any variety of artifacts. (github.com)

• Example SBOM: [https://github.com/microsoft/sbom-tool/releases/download/v0.3.3/win-x64-manifest.spdx.json](https://github.com/microsoft/sbom-tool/releases/download/v0.3.3/win-x64-manifest.spdx.json)

Coming soon:

• **microsoft/CoseSignTool**: CoseSignTool is a platform-agnostic command line application to COSE sign files and validate COSE signatures. CoseParser is a library of functions for COSE signing and validation for use by .NET applications. (github.com)

• sbom-tool and CoseSignTool available soon in the ADK and WDK
  • [Download the Windows Driver Kit (WDK) - Windows drivers | Microsoft Learn](https://docs.microsoft.com/en-us/windows/developer/downloads/download-the-windows-driver-kit-wdk)
  • [Download and install the Windows ADK | Microsoft Learn](https://docs.microsoft.com/en-us/windows/developer/downloads/download-and-install-the-windows-adk)