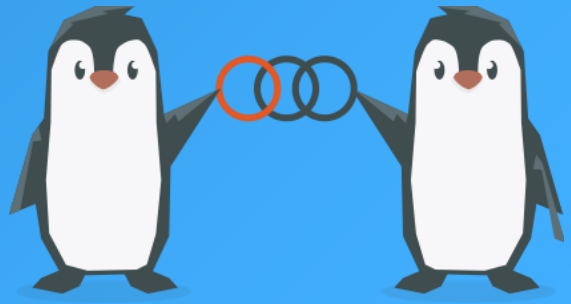
OpenChain SBOM sg #6

2025-02-26

THE LINUX FOUNDATION

OPENCHAIN



Anti-Trust Policy Notice

- Linux Foundation meetings involve participation by industry competitors, and it is the intention of the Linux Foundation to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.
- Examples of types of actions that are prohibited at Linux Foundation meetings and in connection with Linux Foundation activities are described in the Linux Foundation Antitrust Policy available at http://www.linuxfoundation.org/antitrust-policy. If you have questions about these matters, please contact your company counsel, or if you are a member of the Linux Foundation, feel free to contact Andrew Updegrove of the firm of Gesmer Updegrove LLP, which provides legal counsel to the Linux Foundation.

Code of Conduct

- The Linux Foundation and its project communities are dedicated to providing a harassment-free experience for participants at all of our events, whether they are held in person or virtually. Linux Foundation events are working conferences intended for professional networking and collaboration within the open source community. They exist to encourage the open exchange of ideas and expression and require an environment that recognizes the inherent worth of every person and group. While at Linux Foundation events or related ancillary or social events, any participants, including members, speakers, attendees, volunteers, sponsors, exhibitors, booth staff and anyone else, must not engage in harassment in any form.
- This Code of Conduct may be revised at any time by The Linux Foundation and the terms are non-negotiable. Your registration for or attendance at any Linux Foundation event, whether it's held in person or virtually, indicates your agreement to abide by this policy and its terms.

https://events.linuxfoundation.org/about/code-of-conduct/

Recap

Shared the base thoughts on SBOM QUALITY and propose to discuss about our challenges we still have.

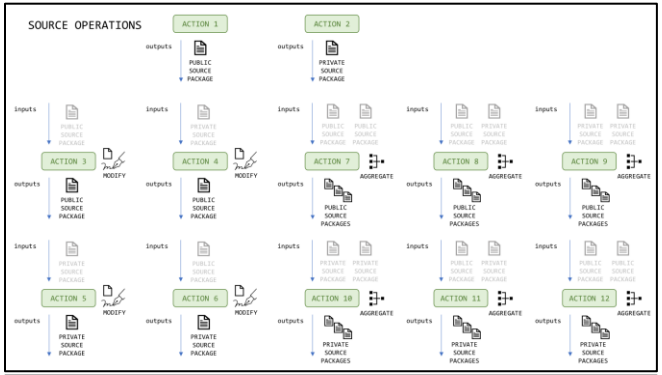
WHAT IS THE HIGH QUALITY SBOM DOCUMENT?

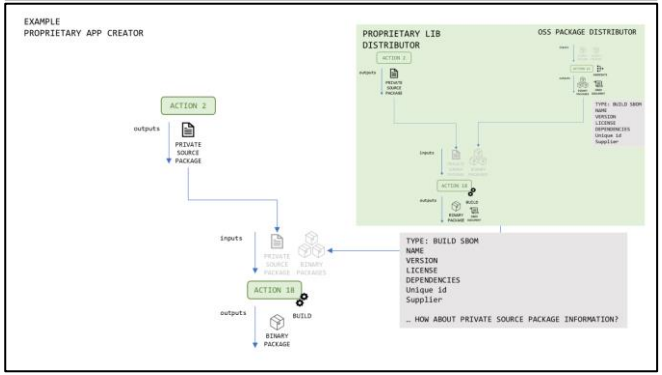
https://github.com/interlynk-io/sbomqs?tab=readme-ov-file#what-is-a-high-quality-sbom

Identify all components of our products from the SBOM document?

Identify all dependencies from the SBOM document?

All components have their versions and ch





CHALLENGE	SOLUTION IDEA
Distributors use different names and versions in many cases, and making it difficult to identify the same package.	Use purl?

Good first Issue

Idea for the SBOM study group

Thinking loud here, maybe this does not make sense...

I was reading https://github.com/OpenChain-Project/OpenChain-JWG/blob/master/subgroups/sbom-sg/meetings/20250203.en.md and started to think that there are thousands of organizations around the world and some have systems and databases that provide, e.g.

- A database of all 3rd party components
- A database of all products and product versions that the organization produces
- A linkage between 3rd party components and products and versions
- A linkage between product versions and 3rd party components needed to build those products (versions)

Now obviously each 3rd party component in the DB comes with metadata and depending on the organization they may put more emphasis on different metadata.

The idea that I started to think is:
Would it make sense to create a **high-level blueprint** of a system that would enable an organization to implement a system that enables them to

- ingest SBOM documents from upstream providers with required metadata, then
- parse and store the SBOM data in the DB (original SBOM document would of course be stored as well)
- maybe add more metadata as needed, e.g. is the component deprecated, project health metrics, etc.
- what to do if the **same** (?) 3rd party component is coming, but with (slightly) different metadata, but for example the PURL may be the same with the already stored 3rd party component
- add new vulnerability data when new vulnerabilities are discovered
- the organization would of course add its own development to the DB
- generate SBOM documents in different formats (SPDX, CDX, Excel, ...)
- generate VEX documents in different formats
- generate SBOM documents with embedded VEX document

For the OpenChain Project, this may become too close to **how**, but maybe we could think of higher level **what** specification and requirements for such a system.

Create sub-issue ☺ ❤ 1

Thanks Jari-san!!

Will convert to the discussions

Today's Topics

https://openchainproject.org/news/2025/02/20/sbom-quality-global-japan

We discussed (and are still discussing) the current challenges of SBOM quality and ideas for solutions within the Japanese community.

I would like to share some of the results.

Full Analysis from Japan SBOM Sub-Group:

Shared Issues

1. Lack of Accuracy in SBOM Information
   There is a lack of consistent and accurate information about package details such as package names, versions, and supplier names because different departments or providers use varying formats.
   - Differences in Package Name Notation
     Even for the same component, different vendors or departments might list slightly different names. For example, one company might mix the official name with abbreviations, include spelling mistakes, or vary between uppercase and lowercase letters, making it difficult for the recipient to consistently identify the component in the SBOM.
   - Differences in Version Information Notation
     For the same component, one source might list "1.0.0" while another might use "v1.0" or other formats, resulting in inconsistent version representations that can cause ambiguity.
   - Variations in Supplier Names
     Similarly, supplier names might vary. Official company names, abbreviations, or even outdated names that do not reflect mergers or acquisitions can be present, leading to an overall lack of consistency.
2. Inconsistency of Component Granularity
   When receiving SBOMs from multiple suppliers, one SBOM might provide detailed information at the file level, while another might compile data at the package level. This inconsistency in granularity requires the recipient to normalize the data.
3. Insufficient Source Information When Binaries Are Provided
   When receiving a binary along with its SBOM, detailed information about the source files (such as file listings, hash values, change histories, etc.) is often missing, making it challenging to manage vulnerabilities or assess risks.
   - Listing of Source Files
     A list detailing which source files are included in the component.
   - Hash Values of Source Files
     Hash values (e.g., SHA-256) for each file to prevent tampering and ensure integrity.
   - License Information
     Information about the license and copyright associated with each source file.
   - Patch or Diff Information
     Details on modifications or patches applied to the source code.
   - Build Environment and Configuration Information
     Data regarding the build environment, such as which compilers or build options were used.
   - Dependency Information
     Details about other libraries or components that this source code depends on.
4. Lack of Integration with Vulnerability Information
   There are cases where the component or library information listed in the SBOM is not adequately correlated

Interesting Challenge no.1

2. Inconsistency of Component Granularity

   When receiving SBOMs from multiple suppliers, one SBOM might provide **detailed information at the file level**, while **another might compile data at the package level**. This inconsistency in granularity requires the recipient to normalize the data.

https://github.com/OpenChain-Project/OpenChain-JWG/blob/master/subgroups/sbom-sg/meetings/20250203.en.md

Proposed Solutions and Automation Considerations

2. Inconsistency of Component Granularity
   - **Checkpoints**
     - Identification of the granularity used in each SBOM (file level, package level, etc.)
     - Differences in the level of detail provided
   - **Proposed Solutions (Processes/Mechanisms)**
     - Clarify the “desired granularity” either industry-wide or internally, and adopt a unified format with corresponding guidelines
     - Build a conversion tool (intermediate tool) to normalize and standardize the granularity of the SBOM data received
   - **Automation Feasibility**
     - Automatable: It is possible to create parsers or conversion scripts that transform various SBOM formats into a unified one.
     - Manual Verification: In cases where the rules are complex, final verification might require human judgment.

Interesting Challenge no.2

3. Diversity of Information and Lack of Standardization
   - To properly manage licenses and track vulnerabilities, there may be **additional information required beyond the NTIA minimum elements** (e.g., build options or environmental details).
     However, there is no standard determining in which situations which information is necessary or how each field should be populated.
   - Some tools lack adequate support for specific programming languages or package managers, resulting in variations in the information collected and the fields used to record it, **depending on the tool**.
   - In the supply chain, packages may be customized or repackaged, which can **lead to inconsistencies with the original build information** and result in inaccuracies regarding dependencies and details.

https://github.com/OpenChain-Project/OpenChain-JWG/blob/master/subgroups/sbom-sg/meetings/20250217.en.md

Proposed Solutions and Automation Considerations

3. Diversity of Information and Lack of Standardization
   - **Checkpoints**
     - Check for the presence of NTIA minimum elements as well as additional required information (e.g., build options, environmental details)
     - Confirm differences in field outputs among various tools
     - Ensure consistency with original build information even when repackaging or customization occurs within the supply chain
   - **Proposed Solutions (Process/Mechanism)**
     - Develop guidelines within the industry or community to agree on “mandatory fields” and “additional information”
     - Standardize mapping rules and conversion logic among various tools to automatically consolidate outputs into a standardized format
     - Establish operational rules and audit processes to ensure original build information is not lost during repackaging or customization
   - **Automation Feasibility**
     - Automatable: Convert tool outputs into standardized formats and automatically map and verify collected data against predefined checklists, including suggesting candidates for missing information
     - Manual Verification: Necessary to handle specific project or environment variations and to automatically detect and reconcile discrepancies arising from customizations in the supply chain

Having Similar Challenges or Solution Ideas?

- Please discuss with your friends and share them here!
  - Please let us know with ML or directly.
- Create Issues on the GitHub and continue to discuss here!
  https://github.com/OpenChain-Project/SBOM-sg/issues