Formats & Tooling Working Group

Co-chairs: JC Herz & Kate Stewart
Meeting biweekly since July 2018

- Fridays at 1100 EDT
- https://lists.linuxfoundation.org/mailman/listinfo/ntia-sbom-formats

Adoption of concepts from framing into existing formats and tools that work with those formats.
Agenda

• Workgroup Goals
• Recap of Formats in Use
  • Updated Formats White Paper
• Taxonomy update
• Playbooks
  • Consumer Playbook Overview
  • Supplier Playbook Overview
• First Plugfest feedback
• Future Directions
• Feedback Requests
Formats and Tooling Workgroup Goal

Wrapping up from phase I, we identified for the need for:

- **Tooling**
  - Documenting tooling
  - Identifying tooling gaps ← Plugfest starting to highlight
  - Documenting processes ← Playbooks starting to address
  - Turnkey universal translation tools

Formats and Tooling workgroup is focusing on addressing these items.
Formats and Tooling: Objectives

Identify SBOM Formats in Commercial Use

- SPDX - https://spdx.github.io/spdx-spec/
- CycloneDX - https://cyclonedx.org/docs/latest

Identify Software Identifiers in Commercial Use and Emerging Identifiers

- Common Platform Enumeration - CPE
- Package URLs - PURL
- Software ID tags - SWID tag
- Software Heritage persistant ID - SWHID
Formats and Tooling: Objectives

• Define and categorize criteria for the minimum required information in an SBOM from Framing Definitions
  • Field definitions
  • Data extensions for provision of additional/external/deeper information

• Enable translation between SBOM formats
  • See emerging tooling & taxonomy

• Create Playbooks for Generation and Consumption of SBOM
  • Supplier Playbook - draft release: https://docs.google.com/document/d/16FwpPX3P0Pd1D82Dp2VmpRnaMWUA-wvfXbL7AIXDthM/edit
  • Consumer Playbook - draft release: https://docs.google.com/document/d/1Ae0l1MDS8m1onS8e8mdVIA9NujzPD0k5j352VIDZr9I/edit

++ Create a reference Corpus of example SBOMs
  • Plugfest to identify Gaps
  • Agree on reference examples to use in each format for others to consult.
## What should a minimum viable SBOM contain?

<table>
<thead>
<tr>
<th>NTIA SBOM Minimum Fields</th>
<th>SPDX</th>
<th>CycloneDX</th>
<th>SWID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supplier Name</td>
<td>(3.5) PackageSupplier: publisher</td>
<td></td>
<td>&lt;Entity&gt; @role (softwareCreator/publisher), @name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component Name</td>
<td>(3.1) PackageName:    name</td>
<td>&lt;softwareIdentity&gt; @name</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unique Identifier</td>
<td>(3.2) SPDXID:         bom/serialNumber and component/bom-ref</td>
<td>&lt;softwareIdentity&gt; @tagID</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version String</td>
<td>(3.3) PackageVersion: version</td>
<td>&lt;softwareIdentity&gt; @version</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component Hash</td>
<td>(3.10) PackageChecksum: hash</td>
<td>&lt;Payload&gt;/../&lt;File&gt; @[hash-algorithm]:hash</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relationship</td>
<td>(7.1) Relationship:   CONTAINS (Nested assembly/subassembly and/or dependency graphs)</td>
<td>&lt;Link&gt; @rel, @href</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Author Name</td>
<td>(2.8) Creator:        bom-descriptor:metadata/manufacturer/contact</td>
<td>&lt;Entity&gt; @role (tagCreator), @name</td>
<td></td>
</tr>
</tbody>
</table>

Translating between SBOM Formats & File Types

SwiftBOM: (SPDX(.spdx), SWID(.xml), CycloneDX(.xml,.json))
- Demo at: https://democert.org/sbom/
- Source code at: https://github.com/CERTCC/SBOM/tree/master/sbom-demo

DecoderRing: (SPDX (.spdx), SWID(.xml))
- Source code at: https://github.com/DanBeard/DecoderRIng

SPDX tools: (SPDX (.spdx, .json, .yaml, .rdf, .xml, .xls) )
- Demo at: https://tools.spdx.org/app/
- Source code at: https://github.com/spdx/spdx-online-tools

CycloneDX CLI: (CycloneDX (.xml, .json), SPDX(.spdx))
- Source code at: https://github.com/CycloneDX/cyclonedx-cli
## Updated: Taxonomy for Classifying SBOM Tools

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Produce</td>
<td>Build</td>
<td>SBOM is automatically created as part of building a software artifact and contains information about the build</td>
</tr>
<tr>
<td></td>
<td>Analyze</td>
<td>Analysis of source or binary files will generate the SBOM by inspection of the artifacts and any associated sources</td>
</tr>
<tr>
<td></td>
<td>Edit</td>
<td>A tool to assist a person manually entering or editing SBOM data</td>
</tr>
<tr>
<td>Consume</td>
<td>View</td>
<td>Be able to understand the contents in human readable form (e.g. picture, figures, tables, text.). Use to support decision making &amp; business processes</td>
</tr>
<tr>
<td></td>
<td>Diff</td>
<td>Be able to compare multiple SBOMs and clearly see the differences (e.g. comparing two versions of a piece of software)</td>
</tr>
<tr>
<td></td>
<td>Import</td>
<td>Be able to discover, retrieve, and import an SBOM into your system for further processing and analysis</td>
</tr>
<tr>
<td>Transform</td>
<td>Translate</td>
<td>Change from one file type to another file type while preserving the same information</td>
</tr>
<tr>
<td></td>
<td>Merge</td>
<td>Multiple sources of SBOM and other data can be combined together for analysis and audit purposes</td>
</tr>
<tr>
<td></td>
<td>Tool support</td>
<td>Support use in other tools by APIs, object models, libraries, transport, or other reference sources</td>
</tr>
</tbody>
</table>

Next steps - tool collections

● Google docs today already list tools in the three formats
  ○ SWID: http://tiny.cc/SWID
  ○ SPDX: http://tiny.cc/SPDX
  ○ CycloneDX: http://tiny.cc/CycloneDX

● Plan to shift this over to GitHub to allow a more open process.

● Working Group will develop some transparent governance rules
  ○ Allow anyone to submit tools
  ○ Have some clear assertions for tools to get on the list
Playbooks for using “Tools in Operation”

- Concepts of Operation (CONOPS) for how they can be used
  - Generation and Consumption
  - Different Use Cases
    - Software Lifecycle Management
    - Entitlements
    - Vulnerability Management
  - Different Roles in the Supply Chain
    - Third Party Supplier (OSS, Commercial Software)
    - Integrator
    - First-party Developer (Internal Enterprise DevOps)
    - Procurement
    - Compliance (interface with external certifiers, regulators, insurers)
SBOM Playbook: Consumer Playbook

• Acquisition of SBOM from supplier
• SBOM Ingestion and Parsing
• Software Entity Resolution
• Data Flows into Third Party Processes and Platforms
  • Configuration Management Database
  • Security Operations Center
  • Software Asset Management System
  • Supply Chain Risk and Vendor Management

• Ongoing Monitoring
• IP and Confidentiality Status of SBOMs and Underlying Data
SBOM Playbooks: Supplier Playbook

- Supplier definition includes: commercial vendor, contract developer, open source software supplier developing and maintaining OSS code.
- SBOM production workflow: development pipeline vs. legacy processes
- SBOM scope: What’s in the Box
  - Areas of consensus: single application and its compiled dependencies
  - Still in discussion: external services (SBOM formats can do this)
  - Need for clarity about SBOM coverage: runtime dependencies, container contents
  - As long as extent of coverage is clear (i.e. fields present with “no attestation”), level of detail will ultimately be negotiated between supplier and consumer
- Validation of SBOMs (formats)
- Verification of Components
- Provision of SBOMs to recipients
  - IP Status of SBOMs
Plugfest

Plugfest used common set of projects to generate source and binary SBOMs for in the different formats for the same example from different tools, to aid compare & contrast. “Sweat equity” from 11 producers and 6 consumers.

• First Plugfest:
  • Time v1.9 (a small package being used in most OSes)
  • node-express-realworld-example-app (Small example node.js application)
  • zephyr-v2.5.0/samples/hello_world (hello world for embedded)
  • blinky.ex (blinky for embedded)

• Second Plugfest:
  • Same set + a couple of binaries (possibly a container)
<table>
<thead>
<tr>
<th>Organization</th>
<th>Produce /Consume</th>
<th>Formats</th>
<th>Examples Tried</th>
</tr>
</thead>
<tbody>
<tr>
<td>aDolus - FACT</td>
<td>Produce</td>
<td>SPDX, SWID</td>
<td>Time, Node, Zephyr, Blinky</td>
</tr>
<tr>
<td>Copado/New Context</td>
<td>Produce</td>
<td>SPDX, CycloneDX</td>
<td>Node</td>
</tr>
<tr>
<td>CyBeats</td>
<td>Produce + Consume</td>
<td>SPDX, CycloneDX</td>
<td>Time, Node, Zephyr</td>
</tr>
<tr>
<td>Fortress</td>
<td>Produce</td>
<td>CycloneDX</td>
<td>Time, Node, Zephyr, Blinky</td>
</tr>
<tr>
<td>FOSSology</td>
<td>Produce</td>
<td>SPDX</td>
<td>Time</td>
</tr>
<tr>
<td>LLNL longclaw</td>
<td>Produce</td>
<td>SPDX</td>
<td>Time, Zephyr</td>
</tr>
<tr>
<td>ORT + Scancode</td>
<td>Produce</td>
<td>SPDX, CycloneDX</td>
<td>Time, Node, Zephyr</td>
</tr>
<tr>
<td>sFractal</td>
<td>Produce</td>
<td>SPDX, CycloneDX, SWID</td>
<td>Blinky</td>
</tr>
<tr>
<td>Source Auditor</td>
<td>Produce</td>
<td>SPDX</td>
<td>Time, Node</td>
</tr>
<tr>
<td>Synopsys Black Duck</td>
<td>Produce</td>
<td>SPDX</td>
<td>Time, Node, Zephyr, Blinky</td>
</tr>
<tr>
<td>Wind River</td>
<td>Produce</td>
<td>SPDX</td>
<td>Time</td>
</tr>
<tr>
<td>Organization</td>
<td>Produce /Consume</td>
<td>Formats</td>
<td>Examples Tried</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
<td>---------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Ion Channel</td>
<td>Consume</td>
<td>SPDX</td>
<td>Node</td>
</tr>
<tr>
<td>NYP</td>
<td>Consume</td>
<td>SPDX</td>
<td>Node, Time, Blinky, Zephyr</td>
</tr>
<tr>
<td>NSA</td>
<td>Consume</td>
<td>SPDX</td>
<td>-- (did Schema analysis)</td>
</tr>
<tr>
<td>SW 360</td>
<td>Consume</td>
<td>SPDX</td>
<td>Time, Node, Zephyr, Blinky</td>
</tr>
<tr>
<td>binaire.io</td>
<td>Consume</td>
<td>SPDX, CycloneDX</td>
<td>Time, Node, Zephyr, Blinky</td>
</tr>
</tbody>
</table>

Summary:
- Participants found it useful and wanted to have follow on, with same samples plus binaries.
- Flagged the importance of having some set of naming conventions or practices for SBOMs themselves.
- Participants also discussed challenges around packages vs. files.
- The community seemed to gravitate to JSON for encoding.
- Bugs got fixed in the tools as a result of interactions, and pairwise partnering between producers and consumers were discussed.
- Next Plugfest planned for June, aligning with OASIS OpenC2 event.
Next Steps

- Update 2019 White Paper

- Continue to collect tools. Put a comment in the document, so it can be added.
  - **SWID:** [http://tiny.cc/SWID](http://tiny.cc/SWID)
  - **SPDX:** [http://tiny.cc/SPDX](http://tiny.cc/SPDX)
  - **CycloneDX:** [http://tiny.cc/CycloneDX](http://tiny.cc/CycloneDX)

- Continue population of examples via Plugfests and Comparison
  - Reference corpus of examples illustrated with each format and encodings
  - Planning underway for next Plugfest in June, aiming to add more binaries as starting points.

- Finalize Playbooks
  - **Consumer Playbook Draft:** [https://docs.google.com/document/d/1Ae0l1MDS8m1on58e8mdVIA9NujzPD0k5j352VlDZr9I/edit](https://docs.google.com/document/d/1Ae0l1MDS8m1on58e8mdVIA9NujzPD0k5j352VlDZr9I/edit)
  - **Supplier Playbook Draft:** [https://docs.google.com/document/d/1Ae0l1MDS8m1on58e8mdVIA9NujzPD0k5j352VlDZr9I/edit](https://docs.google.com/document/d/1Ae0l1MDS8m1on58e8mdVIA9NujzPD0k5j352VlDZr9I/edit)

- Collaboration with medical and any new PoCs, provide feedback of gaps to framing
- Document formats and tooling for VEX

Volunteers interested on working on above areas? Feedback on proposed approach?
More Info...

Meetings: Weekly, next meeting scheduled for **May 7 at 11am EST.** Contact leads to be added to meeting invite

Mailing List: [ntia-sbom-formats@linuxfoundation.org](mailto:ntia-sbom-formats@linuxfoundation.org)

Subscribe at: [https://lists.linuxfoundation.org/mailman/listinfo/ntia-sbom-formats](https://lists.linuxfoundation.org/mailman/listinfo/ntia-sbom-formats)

Shared Drive: [https://drive.google.com/drive/folders/1KAQ7AWlWMKcSFnRc_S-7XB76xFRRWLmT](https://drive.google.com/drive/folders/1KAQ7AWlWMKcSFnRc_S-7XB76xFRRWLmT)
Backup Material
SBOMs Examples (Work in Progress)

SPDX
- https://github.com/swinslow/spdx-examples - source & binary examples

CycloneDX
- https://github.com/CycloneDX/sbom-examples - binary examples

SWID
- Time 1.9 from Red Hat distro - binary example
Current SBOM Options Available

**SPDX**

- **Version**: 2.1.5
- **Revision**: 1.8
- **Date**: 2021-01-01
- **File formats**: .xls, .spdx, .rdf, .json, .xml

**CycloneDX**

- **Version**: 1.4
- **Date**: 2021-01-01
- **File formats**: .json, .xml

**SWID**

- **Version**: 1.1
- **Date**: 2021-01-01
- **File formats**: .xml

File formats: .xls, .spdx, .rdf, .json, .yml, .xml

File formats: .json, .xml
# Information Collected per Tool

## Tool Template

<table>
<thead>
<tr>
<th>Support</th>
<th>Produce?, Consume?, Transform?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td>Website: Source:</td>
</tr>
<tr>
<td>Installation instructions</td>
<td></td>
</tr>
<tr>
<td>How to use</td>
<td></td>
</tr>
<tr>
<td>Versions Supported</td>
<td></td>
</tr>
</tbody>
</table>

## Example: FOSSology

<table>
<thead>
<tr>
<th>Support</th>
<th>Produce (Analyze, Edit), Consume(View, Diff, Import), Transform(Translate, Merge, Tool Support)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>FOSSology is an open source license compliance software system and toolkit allowing users to run license, copyright and export control scans from a REST API. As a system, a database and web UI are provided to provide a compliance workflow. As part of the toolkit multiple license scanners, copyright and export scanners are tools available to help with compliance activities.</td>
</tr>
<tr>
<td>Location</td>
<td>Website: <a href="https://www.fossology.org/">https://www.fossology.org/</a></td>
</tr>
<tr>
<td>Installation instructions</td>
<td><a href="https://github.com/fossology">https://github.com/fossology</a></td>
</tr>
<tr>
<td>How to use</td>
<td><a href="https://www.fossology.org/get-started/">https://www.fossology.org/get-started/</a></td>
</tr>
<tr>
<td>Versions Supported</td>
<td>SPDX 2.1, SPDX 2.2</td>
</tr>
</tbody>
</table>
Tool Support for Different SBOM Formats

**SPDX**
- Format Overview
- Format Publishing History
- Tool Classification Taxonomy
- Open Source Tools
  - Apache
  - Eclipse
  - FOSSA
- Proprietary Products
  - CyberTpacK
  - F3D

**CycloneDX**
- Format Overview
- Format Publishing History
- Tool Classification Taxonomy
- Open Source Tools
  - CyboxDX for Java
  - CyboxDX for .NET
  - CyboxDX for Groovy

**SWID**
- Format Overview
- Format Publishing History
- Tool Classification Taxonomy
- Open Source Tools
  - SWID-Tag
  - Open Source SWID Tag Generator

http://tiny.cc/SPDX
http://tiny.cc/CycloneDX
http://tiny.cc/SWID
Areas to Learn: Generalized vs. Industry-Specific Requirements

- Generalized requirements for code: software, firmware, embedded
- Where do SBOM requirements of firmware/embedded diverge from IT?
  - Ex: Auto industry, Energy, Medical devices with firmware and embedded
- Where do SBOM requirements for licensed/proprietary third party components diverge from third party open source components?
- Lessons Learned and Best Practices for SBOM IP
  - Open Formats
  - Content may be delivered under NDA
  - Content must be capable of transfer to final-goods-assembler without copyright restriction
    - Assumption: NDAs carry the weight of confidentiality terms
- Why this matters: SBOM is an intermediary phase of the data
  - Operational requirement for data to be ingested by enterprise processes and platforms
  - Ex: CMDB, SAM, SOC
  - Configuration management can’t become a “derivative work” and function as intended.