UEFI Support for Software Bill of Materials (SBOM)

UEFI 2022 Virtual Summit
September 28, 2022
Meet the Presenters

**Felix Polyudov**
Director of Firmware Core Architecture, AMI

Felix leads Firmware Core Architecture Group at AMI.

**Brian Mullen**
Senior Manager of Software Security, AMI

Brian is AMI's Secure Development Lifecycle expert leading in areas of security best practices for software and firmware development.
Agenda

• SBOM Use Case Review
• SBOM Implementation Approaches
• UEFI SBOM Implementation
• SBOM Ecosystem Suggestions
# SBOM Use Cases

<table>
<thead>
<tr>
<th>INVENTORY TRACKING</th>
<th>SOFTWARE DEPENDENCIES</th>
<th>PROVENANCE</th>
<th>PEDIGREE</th>
<th>VULNERABILITY STATUS</th>
<th>LICENSE ATTRIBUTION</th>
<th>INTEGRITY, AUTHENTICITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Component Name</td>
<td>- Ability to visualize dependencies with unidirectional acyclic graphs</td>
<td>- Software Origination Details</td>
<td>- Details of changes to software</td>
<td>- Provides the ability to detail the state of vulnerabilities in the product at the time the SBOM was created.</td>
<td>- Facilitates license compliance</td>
<td>- Mechanisms are supported to ensure SBOM information is authentic.</td>
</tr>
<tr>
<td>- Vendor Name</td>
<td>- Determine components that are affected</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Version</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Unique Identifier</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# FW approaches to SBOM

<table>
<thead>
<tr>
<th>Method</th>
<th>Benefit</th>
<th>Drawback</th>
<th>Related</th>
</tr>
</thead>
</table>
| SBOM in the Binary         | Not dependent on any other systems to derive complete SBOM therefore SBOM data guaranteed to be available even if author is no longer available. | - Adds size to the binary object  
- Need a tool to extract the SBOM | Embedding coSWID tags in the binary object  
files https://github.com/hughsie/python-uswid |
| SBOM Reference in the Binary | Small size, easy to update                                               | - Need a tool to extract the references  
- Need systems to facilitate fetching BOM for each SWID                  | Embedding coSWID tags in the binary object  
https://github.com/hughsie/python-uswid |
| Measured Reference         | Little to no size added to binary                                        | Need a system to measure the binary  
Need a system to cross-reference the measurement with a DB of SWIDs.  
Need a system to facilitate fetching BOM for each SWID                   | Intel proposes leveraging TPM architecture to implement SBOM:  
https://uefi.org/node/4261  
OCP beginning SBOM discussions this quarter with this approach in focus |
Tags in binaries

• FW structure dependent
  – If transparency is a goal, we should strive for a structurally independent way to extract the SBOM info, store tags in the clear, limit use of proprietary tooling

• For UEFI, granularity with regards to the UEFI FS structure needs to be considered.
  – Per image, per FV, per FFS, per section

• What do tags contain?
  – References to source
  – Binary Identities (name, version, hash)
### Binary Tagging – Methods and Tradeoffs

<table>
<thead>
<tr>
<th>Type</th>
<th>Per Image</th>
<th>Per FFS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Embedded Identifier</strong></td>
<td>Limited value:</td>
<td>Better:</td>
</tr>
<tr>
<td></td>
<td>• As rom images names and versions already available for released binaries in most cases. IBV’s don’t release rom image binaries nor do silicon vendors.</td>
<td>• Allows for inventory enablement. Can see who in the supply chain last modified the ingredient.</td>
</tr>
<tr>
<td>Challenges:</td>
<td></td>
<td>Challenges:</td>
</tr>
</tbody>
</table>
|                          | • It is difficult to identify FW. Limited universal naming convention for all possible variants. CPE/PURL exist. | • providing chronological versioning scheme.  
• Business sensitivities – info leakage |
| **Embedded Reference to SBOM** | Best: Provides way to obtain SBOM generated from the build for the binary. | Best:                                                                 |
|                          |                                                                          | • Support edge case of upgrading binaries in FFS without requiring complete rom_image upgrade. |
| Challenges:              |                                                                          | Challenges:                                                           |
|                          |                                                                          | • Higher level implementation effort.  
• Need ecosystem capable of supporting per binary SBOMS.  
• No guarantee the SBOM will available in the future (lives and dies with vendor) |
SBOM Implementation Challenges

- Complicated ecosystem (multiple parties involved) with large portions of content exchange in the source form
- Patches are possible at multiple levels
- Component SBOM that was accurate when it left party A may be inaccurate when it leaves party B
- Variety of preferences for SBOM content and level of granularity
- If SBOM data is provided by humans, how to avoid errors? If it's extracted by the tool, how to ensure it's up to date?

Acronyms used in the diagram are defined at https://edk2-docs.gitbook.io/understanding-the-uefi-secure-boot-chain/glossary
Proposed Solution

• Single responsible party
  – Entity that constructs final ROM image (leaf node in the dependency graph) produces SBOM using tooling/infrastructure from the implementation provider

• Upstream partners provide SBOM for their components

• Mechanism to describe patches on top of the upstream components

• Extensibility/Flexibility (ability to put more data into SBOM; ability to adjust the granularity)

• Combine data extracted from the code base with the manually entered data; support overrides of the code base data
SBOM Construction

Package 1
- Package1.dec
- Package1.sbom.yaml

Package 2
- Package2.dec
- Package2.sbom.yaml

Package N
- PackageN.dec
- PackageN.sbom.yaml

Project Package
- Project.dsc
- Project.sbom.yaml

SBOM Builder

Build system specific component description file

Component SBOM file

SBOM Reference is embedded into system firmware

SBOM Reference Data Structure

CycloneDX SBOM

Command Line Arguments

www.uefi.org
# Sources of SBOM Data

<table>
<thead>
<tr>
<th>Department of Commerce</th>
<th>Minimum Elements of an SBOM</th>
<th>Source of Information (Manual)</th>
<th>Source of Information (Automated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supplier</td>
<td></td>
<td>SBOM Builder command line</td>
<td></td>
</tr>
<tr>
<td>Component Name</td>
<td></td>
<td>Component \code{sbom.yml}: name</td>
<td>\code{.dec file: PACKAGE_NAME}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\code{.inf file: BASE_NAME}</td>
</tr>
<tr>
<td>Version of the Component</td>
<td></td>
<td>Component \code{sbom.yml}: version</td>
<td>\code{.dec file: PACKAGE_VERSION}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\code{.inf file: VERSION_STRING}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Commit ID</td>
</tr>
<tr>
<td>Unique ID for look-ups</td>
<td></td>
<td>Component \code{sbom.yml}: id</td>
<td>\code{.dec file: PACKAGE_GUID}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\code{.inf file: FILE_GUID}</td>
</tr>
<tr>
<td>Dependency Relationship</td>
<td></td>
<td>Component \code{sbom.yml}: contains</td>
<td></td>
</tr>
<tr>
<td>Author of SBOM Data</td>
<td></td>
<td>Component \code{sbom.yml}: owner</td>
<td></td>
</tr>
<tr>
<td>Timestamp</td>
<td></td>
<td>SBOM Builder</td>
<td></td>
</tr>
</tbody>
</table>
Beyond Required Elements

• Sbom.yaml file may contain additional relevant data:
  – Component license
  – List of patches applied to upstream components
  – Subset of upstream component used by the FW

• New elements can be added as necessary
  – Extensible format: YAML
Sample SBOM YAML files

# Imported open-source code
name: edk2
id: 6C5BD3EB-AA1F-4DD1-8EE4-963BF4A68079
version: edk2_23
owner: ami
contains:
edk2:
  url: https://www.tianocore.org
  license: BSD-2-Clause-Patent
  version: edk2-stable202205
  scope:
    - MdePkg/*
    - MdeModulePkg/*
patches:
  DriverXOverrun:
    id: CVE-2022-12345, bz1234
    scope:
      - MdeModulePkg/Universal/DriverX/DriverX.c
    comments: Fixed using bz1234 patch

# Native feature package
name: FeatureComponent1

# The rest of the data is extracted from the
# component description file

# Silicon vendor reference code
name: Isv1RcPkg
id: DB3383F3-D696-459D-B60B-8D0754A4B61C
version: Isv1RcPkg_12
owner: ami
contains:
  IsvRc:
    license: Isv1
    scope:
      - *
    version: 2.22.47.31
  Fsp:
    license: Isv1
    scope:
      - *
    version: 1.23
Aptio V SBOM Report
# SBOM Reference Data Structure

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UINT64</td>
<td>Signature</td>
<td>SIGNATURE_64('F', 'W', '_', 'S', 'B', 'O', 'M', 'R')</td>
</tr>
<tr>
<td>UINT16</td>
<td>Size</td>
<td>Total size of the SBOM Reference Structure in bytes</td>
</tr>
<tr>
<td>UINT16</td>
<td>Version</td>
<td>Version of the SBOM Reference Structure</td>
</tr>
<tr>
<td>UEFI_GUID</td>
<td>SbomId</td>
<td>16-byte SBOM identifier of a given firmware’s static configuration of code</td>
</tr>
<tr>
<td>UINT8</td>
<td>SupplierNameSize</td>
<td>Size in bytes of the SupplierName field.</td>
</tr>
<tr>
<td>CHAR8[VendorNameSize]</td>
<td>SupplierName</td>
<td>FW. Supplier. NULL-terminated string. *See iana.org link in the Resources slide</td>
</tr>
<tr>
<td>UINT8</td>
<td>ProjectNameSize</td>
<td>Size in bytes of the ProjectName field.</td>
</tr>
<tr>
<td>CHAR8[ProjectNameSize]</td>
<td>ProjectName</td>
<td>The Project name as a NUL-terminated ASCII string.</td>
</tr>
<tr>
<td>UINT8</td>
<td>FirmwareVersionSize</td>
<td>Size in bytes of the FirmwareVersion field.</td>
</tr>
</tbody>
</table>
Embedding the SBOM Ref Data

<table>
<thead>
<tr>
<th>UEFI FIRMWARE</th>
<th>FV_00</th>
<th>FV_01</th>
<th>FV_02</th>
<th>FV_03</th>
<th>FV_04</th>
<th>FV_05</th>
<th>FV_06</th>
<th>FV_07</th>
<th>FV_08</th>
<th>FV_09</th>
<th>FV_10</th>
<th>FV_11</th>
<th>FV_12</th>
<th>FV_13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FV_13</th>
<th>FFS_00</th>
<th>FFS_01</th>
<th>FFS_02</th>
<th>...</th>
<th>...</th>
<th>FFS_15</th>
<th>...</th>
<th>FFS_56</th>
<th>FFS_57</th>
<th>FFS_58</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>FFS_15</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBoM Section</td>
</tr>
</tbody>
</table>
SBOM Advertisement and Discovery

• If you have the FW binary, you need to devise a way to get the SBOM given only the binary
• If you are relying on SBOM references, the solution should stand the test of time
  – Avoid references that could become stale or obsolete due organizational vacillations
    • Links to vendor SBOM servers – Bad
    • Generic reference to vendor with generic service that maps to vendors to SBOM servers
    • Open, centralized SBOM repo (think NVD or ICAN)
    • Decentralized solution?
Example SBOM Ecosystem

Platform Vendor

UEFI Build flow

Source code

SBOM Yaml

SBOM Build

FW Build

SBOMs

FW Images

http://www.vendor.com/sbom/

One-time registration

UEFI Build flow

SBOM Ref Data Extraction flow

SBOM Tag

SupplierName = 'Vendor Name'
SbomGuid = <SBOM_GUID>
FirmwareVersion = <FW_VERSION>

http://www.vendor.com/sbom/<SBOM_GUID>/FW_VERSION

SBOM Tag Extractor Tool

SBOM Locator Service

SBOM server addr

http://www.vendor.com/sbom/

SBOM Retrieval

SBOM Download Flow

Requestor

Implement SBOM Use Cases

Platform Vendor

SBOMs

SBOM

Public Domain

http://www.uefi.org

www.uefi.org
Next Steps

- SBOM Integration:
  - Downstream partners (OEM/ODM/CSPs)
  - Upstream partners (Silicon Vendors)
Call-to-Action

Contact-Us:
- Demos and Product Updates

Upcoming:
- SBOM Demo: OCP Global Summit 2022 (October)
- Production-ready SBOM: Q1 2023
Resources

Good intro to SBOM use cases:
• https://www.youtube.com/watch?v=PNYyMpUey7Y (OWASP SBOM use cases)

Executive Order Related: Why we have to do it:
• https://www.nist.gov/itl/executive-order-14028-improving-nations-cybersecurity

Advertisement and discover:
• Standard Vendor Names: https://www.iana.org/assignments/enterprise-numbers/enterprise-numbers

Methods/Tools for associating SBOMs with binaries:
• https://github.com/hughsie/python-uswid (LVFS/ Redhat/Richard Hughes' embedded coSWID tags solution)
• https://uefi.org/node/4261 (Intel's approach with TPM/RIM)
Questions?
Thanks for attending the UEFI 2022 Virtual Summit

For more information on UEFI Forum and UEFI Specifications, visit http://www.uefi.org