

presented by



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



 INVENTORY TRACKING	 SOFTWARE DEPENDENCIES	 PROVENANCE	 PEDIGREE	 VULNERABILITY STATUS	 LICENSE ATTRIBUTION	 INTEGRITY, AUTHENTICITY
<ul style="list-style-type: none">- Component Name- Vendor Name- Version- Unique Identifier	<ul style="list-style-type: none">- Ability to visualize dependencies with unidirectional acyclic graphs- Determine components that are affected	<ul style="list-style-type: none">- Software Origination Details	<ul style="list-style-type: none">- Details of changes to software- Vulnerability remediations	<ul style="list-style-type: none">- Provides the ability to detail the state of vulnerabilities in the product at the time the SBOM was created.	<ul style="list-style-type: none">- Facilitates license compliance	<ul style="list-style-type: none">- Mechanisms are supported to ensure SBOM information is authentic.

FW approaches to SBOM



Method	Benefit	Drawback	Related
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



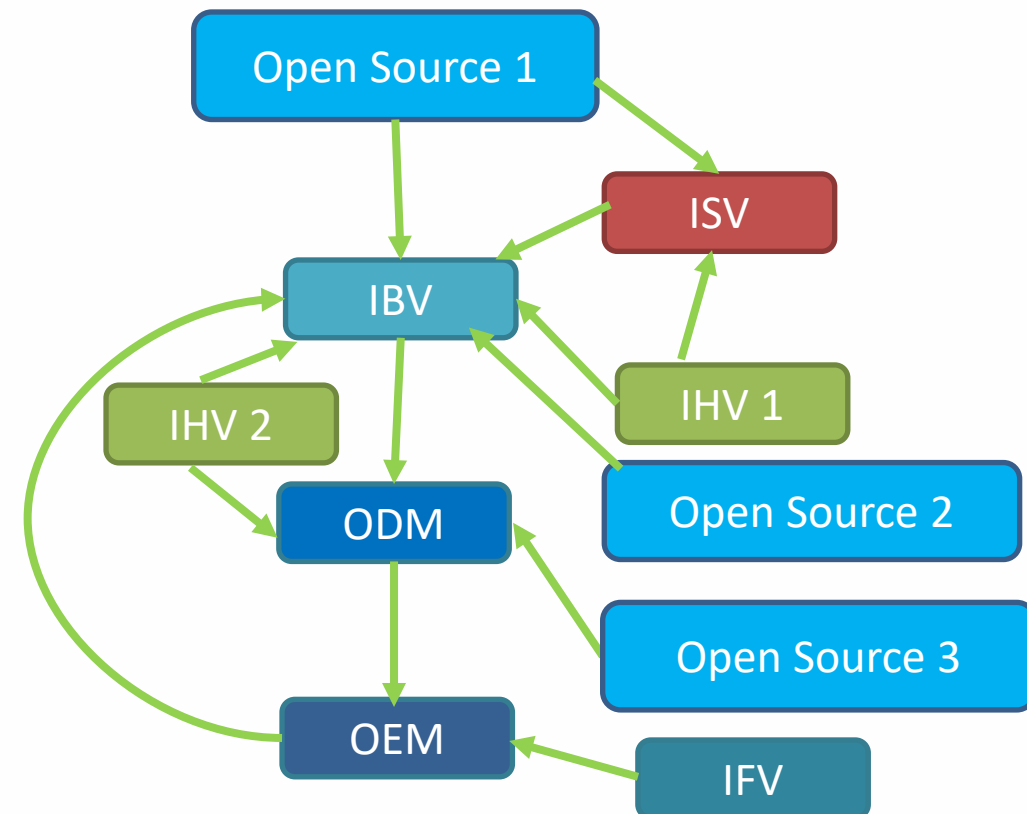
Granularity Type	Per Image	Per FFS
Embedded Identifier	Limited value: <ul style="list-style-type: none"> As rom images names and versions already available for released binaries in most cases. IBV's don't release rom image binaries nor do silicon vendors. 	Better: <ul style="list-style-type: none"> Allows for inventory enablement. Can see who in the supply chain last modified the ingredient.
	Challenges: <ul style="list-style-type: none"> It is difficult to identify FW. Limited universal naming convention for all possible variants. CPE/PURL exist. 	Challenges: <ul style="list-style-type: none"> providing chronological versioning scheme. Business sensitivities – info leakage
Embedded Reference to SBOM	Best: <ul style="list-style-type: none"> Provides way to obtain SBOM generated from the build for the binary. 	Best: <ul style="list-style-type: none"> Support edge case of upgrading binaries in FFS without requiring complete rom_image upgrade.
	Challenges: <ul style="list-style-type: none"> Needs ecosystem so SBOM can be fetched with the SBOM reference. --- Needs SCA phase at build time to assemble ingredients of build 	Challenges: <ul style="list-style-type: none"> 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)

Practical SBOM Implementation



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?



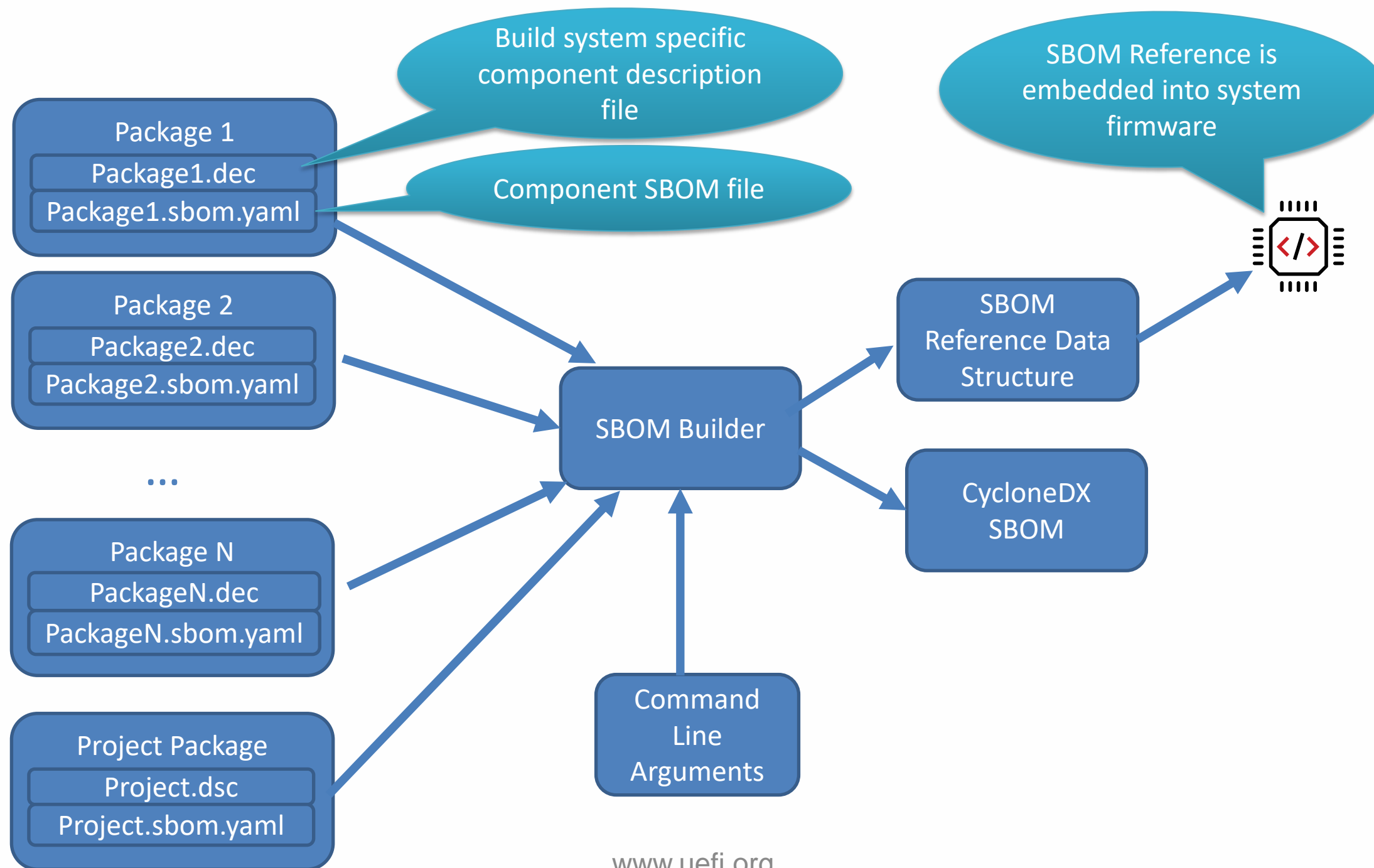
Practical SBOM Implementation



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



Sources of SBOM Data



Department of Commerce Minimum Elements of an SBOM	Source of Information (Manual)	Source of Information (Automated)
Supplier	SBOM Builder command line	
Component Name	Component sbom.yml: name	<ul style="list-style-type: none"> • .dec file: PACKAGE_NAME • .inf file: BASE_NAME
Version of the Component	Component sbom.yml: version	<ul style="list-style-type: none"> • .dec file: PACKAGE_VERSION • .inf file: VERSION_STRING • Commit ID
Unique ID for look-ups	Component sbom.yml: id	<ul style="list-style-type: none"> • .dec file: PACKAGE_GUID • .inf file: FILE_GUID
Dependency Relationship	Component sbom.yml: contains	
Author of SBOM Data	Component sbom.yml: owner	
Timestamp	SBOM Builder	



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

The screenshot shows a JSON viewer interface with two tabs: 'Viewer' and 'Text'. The 'Viewer' tab is active, displaying a tree structure of the SBOM report. The root node is 'JSON', which contains several fields: 'bomFormat' (CycloneDX), 'specVersion' (1.4), 'metadata', 'components', and 'dependencies'. The 'metadata' node contains 'tools' (with a sub-node '0' containing vendor, name, and version) and 'timestamp'. The 'components' node contains three sub-nodes (0, 1, 2) representing different components. The 'dependencies' node contains one sub-node (0) with a 'ref' field.

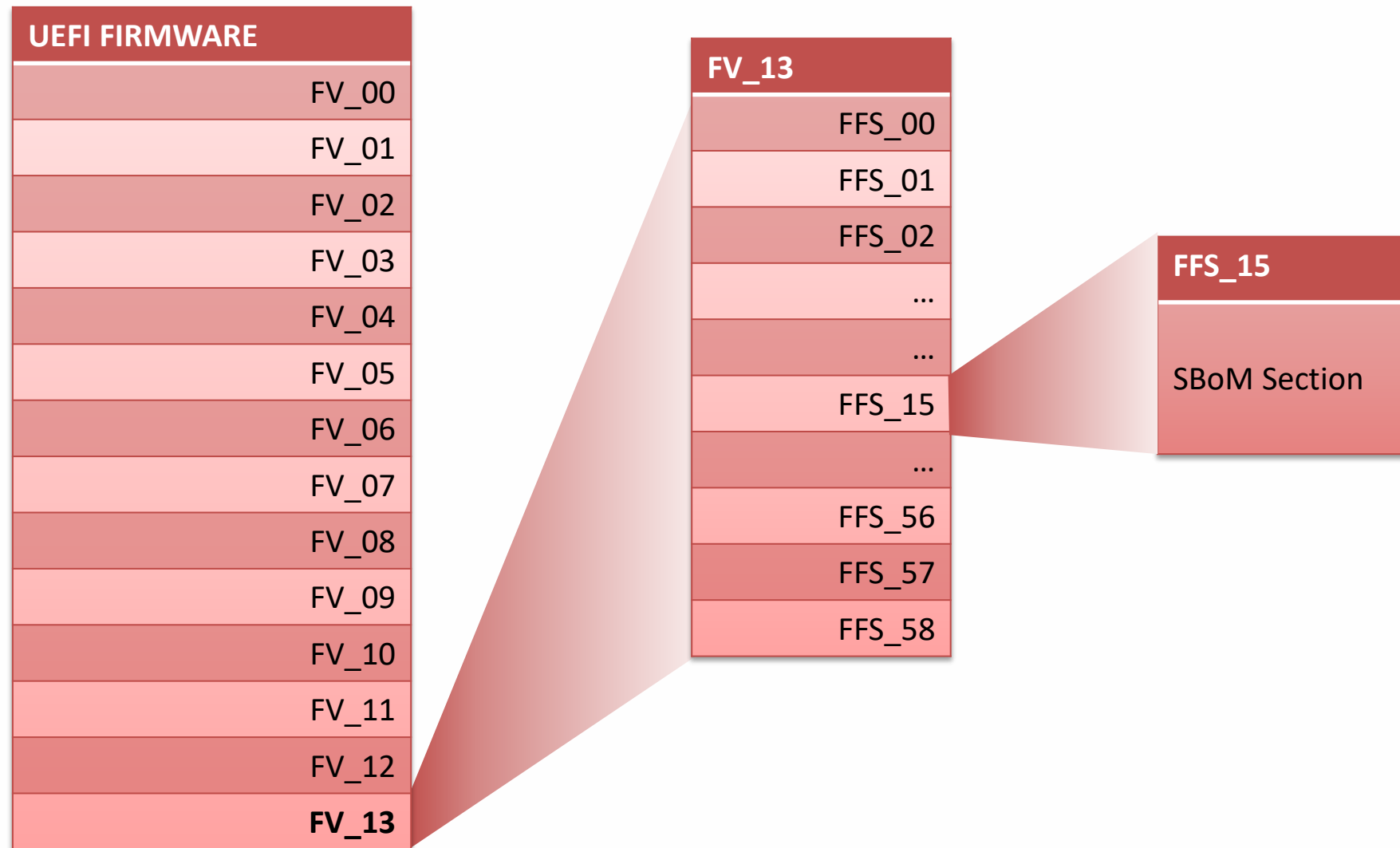
Name	Value
bomFormat	"CycloneDX"
components	...
dependencies	...
metadata	...
serialNumber	"urn:uuid:4c08cea8-76f1-470d-b68f-4644464510e2"
specVersion	"1.4"
version	1

SBOM Reference Data Structure



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

Embedding the SBOM Ref Data

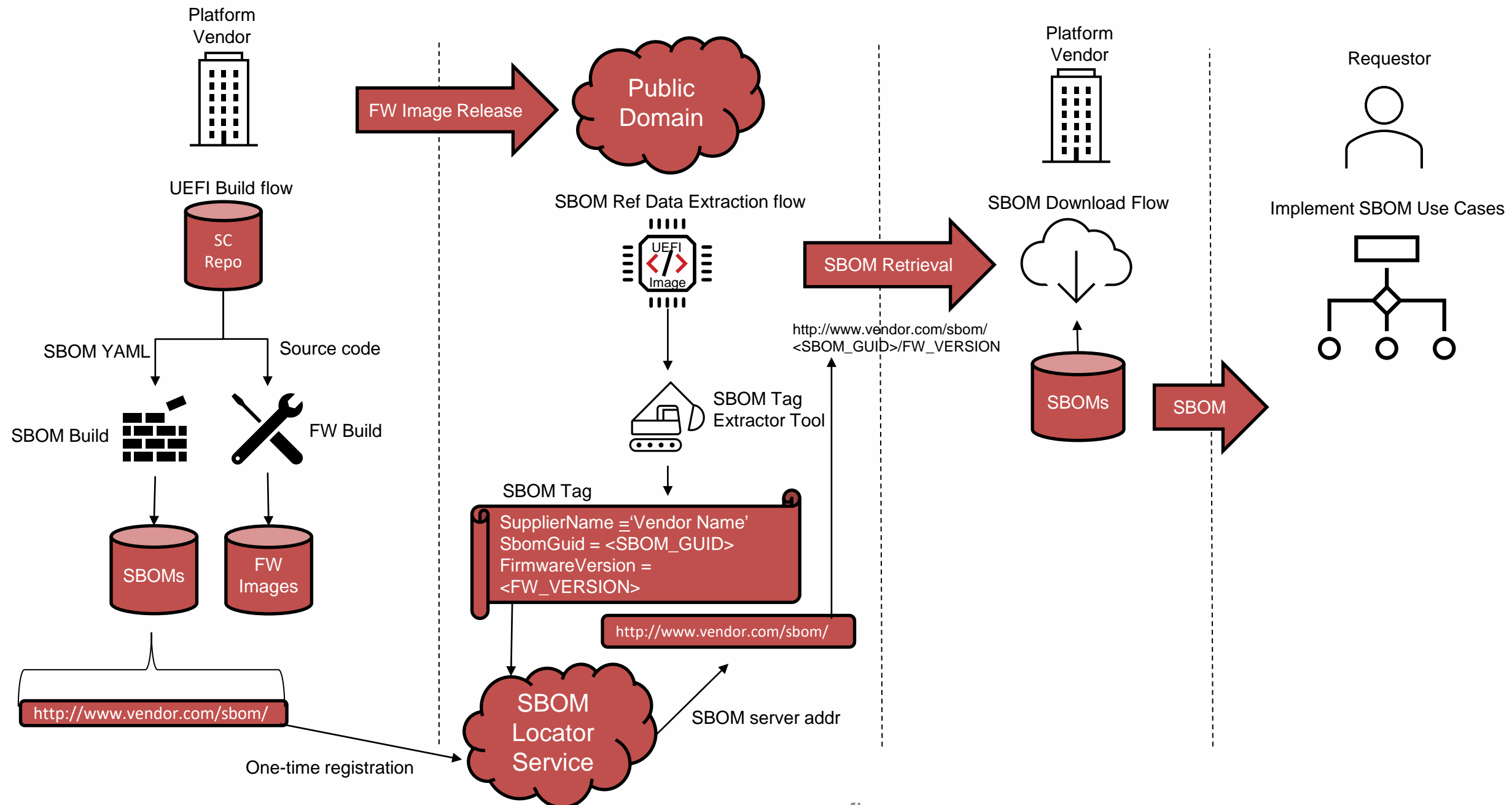


SBOM Advertisement and Discovery



- 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



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>
- https://www.ntia.doc.gov/files/ntia/publications/sbom_minimum_elements_report.pdf

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://www.ietf.org/archive/id/draft-ietf-sacm-coswid-21.txt>
- <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>

presented by

