Building SBOM for System Firmware

Kelly Bryant, CPO, AMI
Brian Mullen, Senior Manager of Software Security, AMI

May 17, 2022
Disclaimer
This publication contains proprietary information that is protected by copyright. No part of this publication may be reproduced, transcribed, stored in a retrieval system, translated into any language or computer language, or transmitted in any form whatsoever without the prior written consent of the publisher, AMI. AMI retains the right to update, change, modify this publication at any time, without notice.

For Additional Information
Please visit ami.com for additional information.

Limitations of Liability
In no event shall AMI be held liable for any loss, expenses, or damages of any kind whatsoever, whether direct, indirect, incidental, or consequential, arising from the design or use of this product or the support materials provided with the product.

Limited Warranty
No warranties are made, either express or implied, with regards to the contents of this work, its merchantability, or fitness for a particular use. AMI assumes no responsibility for errors and omissions or for the uses made of the material contained herein or reader decisions based on such use.

Trademark and Copyright Acknowledgments
Copyright ©2022 AMI. All Rights Reserved.
ami.com
All product names used in this publication are for identification purposes only and are trademarks of their respective companies.
Agenda

• AMI Open-Source Initiatives
• SBOM Overview
• Firmware Supply Chain Challenges
• Strategy
• PoC
• Call-to-Action
AMI IS EMBRACING OPEN SOURCE

Driving innovation through open-source

- Encourage adoption of industry standard solutions
- Develop with a community-first approach
SBOM Overview
Software Bill of Materials (SBOM)

What:
A Software Bill of Materials (SBOM) is a formal record containing the details and supply chain relationships of various components -NTIA

Why:
Compliance: Cybersecurity EO of 2021
Risk Mitigation: Ripple20 – Treck IP stack
## SBOM Capabilities

<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>- Details of changes to software</td>
<td>- Provide the ability to detail the state of vulnerabilities in the product at the time the SBOM was created.</td>
<td>- Mechanisms are supported to ensure SBOM information is authentic.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Vendor Name</td>
<td>- Determine components are affected</td>
<td>- Vulnerability remediations</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>
Firmware Supply Chain Challenges
Challenges

• Tools and processes still in nascent form
  – Universally adopted software naming convention – PURL?
• SCA challenges
• Patch management
• Vulnerability management
  – Needs to be modern and automated (no more spreadsheets and emails)
• How to begin such an effort, get traction, and build momentum?
• Ubiquity of SBOM - We need build tools that are just as capable of generating an SBOM for source code as they are capable of generate binaries from the code.
• How to handle transparency for binary objects?
  – How to get an SBOM that corresponds to the binary
# Transparency via SBOM

Software traverses the supply chain in the form of source code and binaries. An SBOM ecosystem must support the ability to provide an SBOM for the binary or source code representation of the SW.

Consider the flowing approaches:

<table>
<thead>
<tr>
<th>Method</th>
<th>Benefit</th>
<th>Drawback</th>
<th>Related</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store Complete SBOM in the binary</td>
<td>Not dependent on any other systems to derive complete SBOM</td>
<td>Adds size to the binary object</td>
<td>Embedding coSWID tags in the binary object files <a href="https://github.com/hughsie/python-uswid">https://github.com/hughsie/python-uswid</a></td>
</tr>
<tr>
<td>Store a reference to an SBOM in the binary</td>
<td>Small size, easy to update</td>
<td>Need a system to extract SWIDs Need systems to facilitate fetching BOM for each SWID</td>
<td>Embedding coSWID tags in the binary object <a href="https://github.com/hughsie/python-uswid">https://github.com/hughsie/python-uswid</a></td>
</tr>
<tr>
<td>Measured reference</td>
<td>Little to no size added to binary</td>
<td>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</td>
<td>Intel leverages TPM architecture to implement SBOM: <a href="https://uefi.org/node/4261">https://uefi.org/node/4261</a></td>
</tr>
</tbody>
</table>
Strategy
Strategy

• Crawl, Walk then Run

• Identify which use cases matter to the supply chain
  − Start with use cases needed by current organization
  − Create PoCs and socialize
  − Pull in supply chain partners to implement a broader industry wide PoC
    • Sync up on tooling/formats
    • Discover and prioritize use cases of supply chain partners

• Focus on the ability to accurately identify the ingredients in your FW first
  − Good SCA uses snippet checking to find traces of OSS followed by manual investigation to identify versions of that OSS
  − Need to modernize PSIRT/VMS

**PoC: Embed SW IDs in a binary**

<table>
<thead>
<tr>
<th>UEFI BIOS FIRMWARE</th>
<th>FV_13</th>
<th>FFS_15</th>
</tr>
</thead>
<tbody>
<tr>
<td>FV_00</td>
<td>FFS_00</td>
<td>Section_00</td>
</tr>
<tr>
<td>FV_01</td>
<td>FFS_01</td>
<td>Section_01</td>
</tr>
<tr>
<td>FV_02</td>
<td>FFS_02</td>
<td>Section_02</td>
</tr>
<tr>
<td>FV_03</td>
<td>...</td>
<td>SBoM</td>
</tr>
<tr>
<td>FV_04</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>FV_05</td>
<td>FFS_15</td>
<td></td>
</tr>
<tr>
<td>FV_06</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>FV_07</td>
<td>FFS_56</td>
<td></td>
</tr>
<tr>
<td>FV_08</td>
<td>FFS_57</td>
<td></td>
</tr>
<tr>
<td>FV_09</td>
<td>FFS_58</td>
<td></td>
</tr>
<tr>
<td>FV_10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FV_11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FV_12</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FV_13</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Firmware Volume information

Firmware File System information (DXE) In Nested FV

Firmware File System information (PEI)

SBOM Information

[uSWID]
Product = 'Aptio'
Module = 'IsRecovery'
git-repo = 'git@git.ami.com:bootfirmware/aptiov/bkc/cfl.git'
git-commit = '624605f0a4a883a9ae41fc77078a1c6973bb78a0'

[uSWID-Entity]
[uSWID-Entity.Distributor]
Name = 'One AMI'
RegId = 'ami.com'
extra-roles = ['Licensor', 'Maintainer', 'Software Creator']

[uSWID-Entity.TagCreator]
Name = 'One AMI'
RegId = 'ami.com'
extra-roles = []
Advertisement and Discovery

```
git-repo = 'git@git.ami.com:bootfirmware/aptiov/bkc/cfl.git'
git-commit = '624605f0a4a883a9ae41fc77078a1c6973bb78a0'
```

Eco-centric Automation

PUBLIC REPOS
EDK II, OpenBMC, CoreBoot, Aptio/MegaRAC...

DevOps Framework
- OEM/ODM/CSP Customizations or Extensions
- SCA BOT
- SAST BOT
- DAST BOT
- Auto Build
- Auto Test
- CI/CD Pipeline
- SSDLC Score
- SBOM Creation

TESTED & TRusted FIRMWARE

SI PARTNERS
Silicon-specific code & components

SOC PARTNERS
SOC specific code & components

PLATFORM VENDORS
Platform Porting Code & Components

PLATFORM CONSUMERS
Domain specific code & components

PLATFORM VENDORS/CONSUMERS
- OEMs
- ODMs
- System Integrators
- CSPs
- Telcos
- IOTs
- HPCs

PUBLIC REPOS
- EDK II
- OpenBMC
- CoreBoot
- Aptio/MegaRAC

DevOps Framework
- OEM/ODM/CSP Customizations or Extensions
- SCA BOT
- SAST BOT
- DAST BOT
- Auto Build
- Auto Test
- CI/CD Pipeline
- SSDLC Score
- SBOM Creation

TESTED & TRusted FIRMWARE

An Open SBOM Ecosystem

Binary
1010
1010

Measurement App

International BOM DB

Hash
1010
1010

SBOM

IBOM

Vulnerability Report (CSAF) with Hash

Vulnerability Report (CSAF)

IVEX

SW Binary Providers

SBOMs with Hashes

Build an expanded PoC

- Cross-vendor SBOM consumption tool
  - Silicon Vendors (IHV)
  - IFV
  - ODM
  - OEM
  - Point of Use (PoU)

Contact AMI (ami.com/contact)
Resources

Executive Order Related: Why we must do it:

Ripple20: Why we should do it.
- https://www.jsof-tech.com/disclosures/ripple20/

Who is using SBOM and why:

Good intro to SBOM use cases:
- OWASP SBOM Use Cases: https://www.youtube.com/watch?v=PNYyMgIey7Y

Good info on industry wide proof-of-concepts and much more generic SBOM info
- https://www.cisa.gov/cisa-sbom-rema

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)

Proof of concept
- https://teml.io/env/v1.0.0

VEX

SBOM Tooling Info:
- https://cyclonedx.org/tool-center/
- https://spdx.dev/resources/Tools/
Thank you!
VEX + SBOM example

- Software includes a vulnerable component
- SW supplier determines that the vuln doesn’t affect the built software
  - E.g., relevant code isn’t included by compiler
  - E.g., relevant code is present, but not used or exposed
- Supplier issues a VEX with the claim that the component is “not affected” and no action is required
- Consumer integrates SBOM data, vulnerability data, and VEX data to make some risk-based decision