Software Bill of Materials: Progress toward transparency of 3rd party code

Allan Friedman, U.S. Department of Commerce
Art Manion, CERT Coordination Center
Art commutes by bike

• “Torn up grade crossing in bad weather at a low angle, what could possibly go wrong?”
• “Wow it takes longer to heal when you’re over 40.”
Where’s Allan?

• “Flying in the morning of the talk should be fine.”
• “My slides are mainly pictures, surely Art will know what I wanted to say.”
Paying attention vs Checking Email

• The case for transparency
• How transparency can help the software ecosystem
• Why aren’t we doing this already?
• What *is* a Software Bill of Materials?
• How do we do this?
• What next?
Mudge @dotMudge · 27 Aug 2016
If you have a 2013 Mercedes S-class you have libtiff, netcat, and libpcap, pre-installed.
Analogies

**Ingredients:** Corn, Vegetable Oil (Corn, Canola, and/or Sunflower Oil), Maltodextrin (Made from Corn), Salt, Cheddar Cheese (Milk, Cheese Cultures, Salt, Enzymes), Whey, Monosodium Glutamate, Buttermilk, Romano Cheese (Part-Skim Cow's Milk, Cheese Cultures, Salt, Enzymes), Whey Protein Concentrate, Onion Powder, Corn Flour, Natural and Artificial Flavor, Dextrose, Tomato Powder, Lactose, Spices, Artificial Color (Yellow 6, Yellow 5, and Red 40), Lactic Acid, Citric Acid, Sugar, Garlic Powder, Skim Milk, Red and Green Bell Pepper Powder, Disodium Inosinate, and Disodium Guanylate.

**CONTAINS MILK INGREDIENTS.**
Analogies
Analogies (cont’d)
Analogies (cont’d)
A data layer to drive Innovation

Common Vulnerabilities and Exposures
### Supply chain

- Supplier selection
- Supply selection
- Supply vigilance
Three perspectives across the supply chain

• Produce software
• Choose software
• Operate software
Use Cases: Producing software

- Monitor for vulnerabilities in components
- Better manage code base
- Execute white-list or black-list practices
- Prepare and respond to end-of-life contingencies
- Minimize code bloat
- Know and comply with license obligations
- Provide an SBoM for customers
Use Cases: Choosing software

• Identify known vulnerabilities
• More targeted security analysis
• Verify sourcing
• Compliance
• EOL awareness
• Verify some supplier claims
• Understand software integration
• Market signal of secure development process
Use Cases: Operating software

• Vulnerability management
• Better understanding of operational risks
• Real time data on components in assets
• Improved understanding of potential exploitability
• Enable potential non-SW mitigations
So why aren’t we doing this already?
It’s hard.
• Apache2
• Apache Web Server
• Apache
• HTTPd
• HTTPd2
A market failure?
Enter your friends, the Feds
The “multistakeholder” model

- Open to all Stakeholders
- Accountable
- Transparent
- Consensus Driven
- Bottom up process

Open, transparent, consensus based processes that bring together diverse stakeholders can catalyze real progress across the ecosystem.
The “multistakeholder” model
The "multistakeholder" model
What we’re not doing

• Regulation
• Source code disclosure
• Standards development
• Harmonization
• Amplification & routinization
• Extensions & innovation
Making progress

• Clear appreciation across sectors on the potential value of transparency
• Consensus already on
  • The broad scope of the problem
  • Machine-readability of the solution
• “Minimum Viable Identity” (MVI)
Framing

• Conceptual design
• Terminology
• Broad requirements
• Cross-cutting issues

Emerging consensus, or at least temporary acceptance
What is an SBoM?

1. Core information elements: Minimum Viable Identity (MVI)
   • Cryptographic hash (or signature)
2. Other very, very important and useful identify information
   • Supplier (aliases), author, component (aliases), version, relationships
3. Other information necessary for most use cases and applications
   • License, entitlement, vulnerability mapping, formulation, provenance
   • Software components
     • Defined and named by suppliers, at time of delivery (build, package, install, deploy)
     • Hardware not excluded
     • Source code not excluded
Applications

• Intellectual property management
  • Licensing, entitlement
  • Most mature application

• Vulnerability management
  • What components are affected by vulnerabilities?
  • Transitivity – vulnerability is not necessarily exposure or exploitability

• High assurance
  • Provenance, pedigree, formulation, integrity, chain of custody

• Economic benefits of supply chain hygiene
Selected SBoM Elements

• No SBoM without MVI
Intellectual Property

- Well-established application
- Licensing, liability, entitlement
Vulnerability Management

- Requires vulnerability mapping to external catalog
- Related technologies and other components helpful for coordinated disclosure
High Assurance

- Critical systems, national defense
- Formulation: How component was built
- Not shown: Provenance, pedigree, chain of custody
SBoM Processes

• Supplier responsibilities
  1. Define self-created components and create SBoMs
  2. Obtain SBoMs from direct, immediate suppliers
  3. Provide collected set of SBoMs to consumers

• Change SBoM when software changes
  • Patch, update, new version

• Change SBoM when other information changes
  • License, new upstream information

• Challenge: Claims about other suppliers’ SBoMs
  • Author and Supplier are different
Terminology

• SBoM (Software Bill of Materials): inventory and associated information in a standardized format
• Inventory: list of components using Minimum Viable Identity
• Author: entity that creates SBoMs
• Supplier: entity that defines and identifies components and creates associated SBoMs
• Consumer: entity that obtains SBoMs
• Component: unit of software defined by a supplier at the time the component is built, packaged, or distributed
Existing Work

• Software Identification Tags (SWID)
  • ISO/IEC 19770-2, NIST (US)
• Software Package Data Exchange (SPDX)
  • Linux Foundation
• Software Heritage
  • Focus on source code
  • Identifiers for Digital Objects
• package URL (purl)
• Common Platform Enumeration (CPE)
• Software Asset Management (SAM)
• Software Composition Analysis (SCA)
• Supply Chain Risk Management (SCRM)
Example: Simple Table

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Supplier</td>
<td>Component</td>
<td>Version</td>
<td>Hash</td>
<td>Includes</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>OpenSSL</td>
<td>OpenSSL</td>
<td>0.9.8a</td>
<td>0x113a8...</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Apache</td>
<td>httpd</td>
<td>1.3.26</td>
<td>0x33af2...</td>
<td>OpenSSL 0.9.8a</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>MDM1</td>
<td>FooPump</td>
<td>4.0</td>
<td>0x44a83...</td>
<td>Apache httpd 1.3.26</td>
<td></td>
</tr>
</tbody>
</table>
Example: namespace:name

org.openssl:"OpenSSL 0.9.8a"
org.apache:"httpd 1.3.26"
com.mdm1:"FooPump 4.0 0x44a83..."
Example: purl

pkg:tgz/org.openssl/OpenSSL@0.9.8a
pkg:tgz/org.apache/httpd@1.3.26?requires=pkg:tgz/org.openssl/OpenSSL@0.9.8a
pkg:device/com.mdm1/FooPump@4.0?hash=0x44a83...&requires=pkg:tgz/org.apache/httpd@1.3.26
Example: SWID

<SoftwareIdentity name="openssl" tagId="openssl/openssl@0.9.8a" version="0.9.8a"/>
<SoftwareIdentity name="apache_httpd" tagId="apache/httpd@1.3.26" version="1.3.26"/>
<Link href="swid:openssl/openssl@0.9.8a" rel="requires"/>
<SoftwareIdentity name="MDM1 FooPump" tagId="MDM1/FooPump@4.0" version="4.0"/>
<Link href="swid:apache/httpd@1.3.26" rel="requires"/>
Example: SPDX

PackageName: openssl
SPDXID: openssl/openssl@0.9.8a
PackageVersion: 0.9.8a

PackageName: apache_httpd
SPDXID: apache/httpd@1.3.26
PackageVersion: 1.3.26
Relationship: openssl/openssl@0.9.8a PREREQUISITE_OF apache/httpd@1.3.26

PackageName: "MDM1 FooPump"
SPDXID: mdm1/foopump@4.0
PackageVersion: 4.0
Relationship: apache/httpd@1.3.26 PREREQUISITE_OF mdm1/foopump@4.0
Example: Graph
## Example: Additional SBoM Data

<table>
<thead>
<tr>
<th></th>
<th>SWID</th>
<th>SPDX</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hash</strong></td>
<td>hash-entry hash-alg-id hash-value</td>
<td>PackageVerificationCode PackageChecksum FileChecksum</td>
</tr>
<tr>
<td><strong>License</strong></td>
<td></td>
<td>LicenseConcluded PackageLicenseDeclared LicenseName</td>
</tr>
<tr>
<td><strong>Entitlement</strong></td>
<td>@entitlementKey</td>
<td></td>
</tr>
</tbody>
</table>
I deleted regid.1991-06.com.microsoft on my other PC, and it boots up to a black screen now. How do I fix this?

Deleted the entire folder, swidtag and all. Like a ****. I have another laptop with this file on it, and I moved it to a USB key so I could replace it on this other PC. But like I said- it boots to a black screen and I can’t see anything or do anything. I have an MSI motherboard, I’m not sure how to boot into safemode with a pureblack screen. I can’t change to another user because I don’t have one. Just this single profile, with the folder deleted. Help!

C://ProgramData/regid.1991-06.com.microsoft (File path for folder that got deleted).
Open questions to figure out together
Obstacles to obtaining SBOM data?
Federation

• Vertical slices of solution
  • Automatic updates, package managers

• Centralized authority and collection does not scale
  • NIST (US) Common Platform Enumeration (CPE)
  • NIST (US) National Software Reference Library (NSRL)
  • TagVault (for SWID)

• Distribute effort to suppliers (vendors)
  • Least Cost Avoider
  • Most suppliers are also consumers
Opacity and Translucency

• Suppliers have first-hand knowledge about components they originate and those they directly obtain from an upstream supplier
• What happens when SBoM is not available?
  • Knowledge that there are no further upstream dependencies
  • Lack of such knowledge
  • Third-party claims is fragile design
Mechanisms of sharing SBOM data?

Transparency

Vendors

Customers
Transparency Options

• Include SBoM files with install: SWID, SPDX
  • Constrained storage? CoSWID
• Even more constrained storage? Lookup
• Publication
  • ROLIE Software Descriptor Extension
• Cataloging
Vulnerability vs. Exploitability

Vendors can communicate risk (or the lack thereof) with their customers. We need to enable this process.
High Assurance SBoMs
SBoM for Services

ABSTRACT
Continuous deployment is the software engineering practice of deploying many small incremental software updates into production, leading to a continuous stream of 10s, 100s, or even 1,000s of deployments per day. High-profile Internet firms such as Amazon, Etsy, Facebook, Flickr, Google, and Netflix have embraced continuous deployment. However, the practice has not been covered in textbooks and no scientific publication has presented an analysis of continuous deployment.

Next steps

• Drafts of “minimum viable” by late June for feedback

• After minimum viable:
  • Extending the model
  • Developing and collecting tooling
  • Awareness and adoption
  • Testing ⟷ revision
Testing

• Previous attempt at CERT/CC: Component Relationship Database (CRDb)
  • Neo4j, Sesame, RDF
• Next experiment: Index cards and Sharpie
To recap...

- Tracking third party components can help understand and address a wide range of risks across the entire ecosystem
- An ongoing, open process convened by NTIA is bringing together experts to address:
  - What a Software Bill of Materials is
  - Why it can help across the supply chain
  - How we can implement it
- Get involved in the NTIA process!
  - afriedman@ntia.gov @allanfriedman
  - amanion@cert.org @zmanion