This document was compiled in real time from the multistakeholder discussion at the November 6, 2018 multistakeholder meeting on Software Component Transparency. The notes were recorded live in front of participants. No further attempt to edit or organize the raw notes has been made.

For more information, please go to: https://www.ntia.doc.gov/SoftwareTransparency

Open Questions for discussion

- SBOM elements
- How to share data
- Cloud application of SBOM
- Communicating related data
    - Nutrition Labels
- Differential naming

1. SBOM elements

Framing overview

- Minimum viable product – naming not outside of the SBOM
    - CPE naming conventions or congruent names so that it is likely to be identified with that name
    - Looking at where the source is coming from to identify component
- CPE
    - Map vulnerabilities to vulnerable products
    - After product is fielded, analysts need to forensically discover
        - Too late in the software lifecycle
    - This effort could accelerate identification of software before it's installed
        - Can then be mapped to multiple disciplines
    - CPE inadequate from this perspective
    - Proper identification information
        - SWID and SPDX address at vendor source
        - If multiple software identifiers, there is a multiple mapping problem
        - Single installed software identifier can map more simply
- Library name and library version (or checksum)
    - Vulnerabilities map to versions, not products
- Plural names – alias field
- Software creation at build

    - All still need to be connected up and go back to the source when the source is available
- Names of products change all the time
    - Having hash associated is key
    - Name and version is a good starting point
- Components and running programs down the stream
- Flat vs. hierarchical?
    - Might make product x, but still responsible for everything before it
    - Final SBOM is a superset of other SBOMS
- Vendor name, product name, version string
    - Naming can get more complicated
    - Hash naming is at a later step?
- Big question: what is the technical maturity of the customer?
    - Elite customers want fancy stuff, but the bar is relatively low
- ➔ More adoption to an easier, flatter thing, or gear toward more sophisticated customers?
    - Need to start flat – address the "chicken and egg" problem
    - Current state of practice is flat
        - Flat (no information) vs. non-recursive
        - Recursive without cementing value
    - Can always flatten a hierarchical SBOM, cannot work the other way around

Clarify the definition of "flat"

- All transitive dependencies without semantic relationships between them
- Only describing top level includes: direct dependencies

Is there a value in capturing transitive dependencies?

- Want a model that is capable of modelling the graph of relationships as organizations can provide information
- SPDX and SWID can provide semantic information
- Room for growth

Identifiers

- Not enough to have name of product, vendor, version: need packaging and release
- Some allow functional patches, some don't

Various use cases that need SBOM-like thing

- Lack of vocabulary to talk about the various formats
- Really talking about a handful of different things to support various use cases
    - i.e. license information
- ➔ who are the different stakeholders and what do they actually need?

Some customers care about the chain of custody

Configuration management

- what is the software used by
- what does the software use
- ➔ relates in some contexts

Not a lot of faith in ability to track naming

- put forth by marketing typically
- potentially come up with some lower-order naming convention (for example catalog number)

Flat vs. hierarchical vs transitive

- manufacturer perspective – just want a concise list of what's in the product
- want data transitive, but not semantic

Wouldn't you always have a flat BOM for every component?

- Automatically flat and hierarchical
- ➔ As long as you have a way to link, then you're fine
    - Each package has own SBOM effectively – want to express the relationships of all components

What is valuable to what an attacker sees vs. an operator?

- From an attacker view, the SBOM is flat – independent variable from where the SBOM came from

Open source community – scanners find things that the others might not

Continuous monitoring on all transitive dependencies now

- Can recursively go through that data
- ➔ Not all components, and can't limit

Creator of software should only be responsible for top level

- Potentially undermines the SBOM?
- ➔ Looking for a tool to provide an answer to our mission

If you want to be adopted as an open source, need to put in the right parts

As long as producers are responsible for one tier down, then the continuity should solve itself

- Is it a big lift for everyone to do some situational awareness?
    - Yes – if the component in this application is 1.4, but there is a version that is 2.8, put it in the SBOM so the consumer knows the change between what's possible
        - Allows the ability to pull from more updated application in the future
    - Company service

Some value in local structure, have capacity for first level of hierarchy

- Slightly more curated supply chain to have more continuity

Must be liability associated with supply chain

- If a car manufacturer is not liable (responsible) for 5th party parts, the 4th party won't care to offer the components
- Liability – contractual issue between consumer and producer

What do I need up front vs. what do I need later on?

- Talking about a narrow scope
- Bottom up vs top down
- As long as there is a concise accurate list of components, doesn't matter how we get there

Concise list of components

- Ensure that for any device or component, have a list of subcomponents
    - Trouble when bringing in 3rd party components, figuring out a way to force them to do the same thing

Transitive components – at what point do we say that the person compiling should do one more hop upstream

FDA requires CBOM

- If the producer cannot support that, then the consumer won't buy
- Forces top level that rely on open source to understand the chain and clean up certain projects

Minimum viable product

- Start with the top line, but don't exclude other levels

- Can whatever we create be able to work with the use cases we need

Is there a use case where only including the top line is sufficient for use case?

- Software creator creates BOM, tries to get BOM from all the components you used

The data is out there – if not provided in SBOM, that may be fine and consumers can find it elsewhere

"Where am I attacked" question may not be addressed if only go one level deep. Undermines "am I affected, where am I affected"

- Any component listed, you can get transitive open-source dependencies
- Consumers want the whole thing, but there are other ways to derive it besides SBOM

Proprietary software not declaring its dependencies and does not disclose to your "one layer deep," then the final product is vulnerable

Can the one layer only model be used to be able to delve down into all of the components?

- "am I affected" use case requires every layer

Requirement: provide both flat formats (direct and all the way down), and will satisfy everyone's requirements

Want to ultimately end up with the full graph for most use cases

- Is A just trying to get the B layer, which includes the C layer, or is A trying to get all layers

Attackers don't care if it's hierarchical, care if it's accessible

- The top layer needs the first layer so that they don't look through all components if the problem is the most recent layer
- ➔ Potentially want the bottom layer instead?
- ➔ Full list creates noise

List (flat or transitive) vs. Graph

- Vendor perspective – dependent on SBOM for information
  - If no transparency as an open source software, might break it
  - If commercial component, it will break the chain

Shouldn't confuse the particular piece of information needed vs the way the information is found

- What is in it answer is part of what is possible that's in it
- Need to know who's using the particular supplier

- ➔ Still need collection of information for the entire set

Hierarchy matters to figure out what needs to be patched

- ■ Sometimes not exploitable at pervious levels
- ■ Need to know what component was included in

Customer is not here to tell us what they want

- ■ Be aware of the situation where the customers share what they want
    - o Include libraries and software
    - o Dynamic levels of application

Are customers in the room

Need a crisp, clear taxonomy of the graph vs. list organization

- ■ Need to clarify terminology

Use cases – more general, top level needs

- ■ Medical device field – not patching unless you get validation from the manufacturer
    - o Communication and collaboration
    - o Manufacturer responsible for searching and finding vulnerabilities

Patching without validation is not something that we're supposed to be doing in this industry

- ■ Use other mitigation tactics

2. Information sharing

Every build, packages with a deliverable BOM

- Every release upstream also

Need a deliverable for cloud and non-cloud situations

Lightweight device can have readable pointer for traffic profile

- ■ Can contain a directory that lives in another location

Medical tech vendors and IoT vendors

- ■ Point to an ingredients lists
- ■ Vulnerability updates

Want to point somewhere that indicates a patch and how it modifies the original BOM

- Why not just create a new BOM?

There has to be a way to highlight that a patch is sitting there

- Might want to know whether a patch has been applied to your particular instance

Patch vs. Version?

- Patch happens between version
- Going forward, perhaps every change is a new version

Patch: specifically to address a security vulnerability

- Could benefit from more disciplinary control
- ➔ Would be chaos for customers by renaming every time you do a patch because that's not the way cycles work
    - Historical nomenclature
    - Changing names has many implications

Rolling a new component vs fixed a bug

- New version with fixed bug would create chaos

ACTION ITEM QUESTION: semantic arguments (solved online)

make identifying patches site specific

Do you go down to the atom or capture the molecule?

Structural item: Build number

- Date might be a valuable piece of information
    - Software user may not want BOM to expire beyond a certain amount of time
    - Expiration date on BOM?
        - BOM taken as a specific point in time, but updates take place – new BOM every time

        - "As built:" every time something is built, new BOM produced

Live patch vs. custom patch

- Ambiguity is valuable in certain circumstances

Need a uniquely identifiable identifier of software changes

- Whatever meets the criteria of "we can tell the software changed" is all we should be concerned with right now

3. "Cloud"

On prem SBOM, no changes in how we think about it vs. on prem SBOM is wildly different and need to think about them way differently

- Different use case: Cloud service updating on prem

What are the use cases we're trying to solve?

- NIST: scope of Saas, etc.
- "Am I affected" – every change in stack is a change in an SBOM
    - If they are sutured together, less about what is it, more about how to we access
    - No architecture
    - SBOMS are composable
- Content is well protected regardless of what BOM is
    - Designing security controls based on premise that already compromised
    - Inventory on all threats and components
- ➔ Different enough use case that it should not be considered in this group's scope

Web services – treated as atomic items

- Don't want to have to care what's in them
- Vulnerability management and assessment
    - This is focus

Can we have items where we agree not to weigh in too deeply, but have an opinion on?

- Split decisions in court case – can we have "dissent" positions

- Can outline issues that do not have complete consensus

In context of what concerns are

- More concerned about provider rather than software
    - Malicious insider in company is more dangerous than the software it may use

Is it important that the service provider be able to manage list internally?

- Yes – tells me more likely to hear flaws from good guuys before bad guys
- SBOM allows response time to be much different

Cloud is inevitably something to address but later in the workstream

What is the obligation of cloud providers to provide complete, accurate, and full information of APIs?

- Very different in public cloud vs. traditional

On prem product and software – track it down to patch

- Knowing if the cloud is tracking three layers down doesn't provide much utility

SBOM is useful for specific scope of activity

- As an end user, what can you realistically do
- Demonstration that SBOM is "there" is not the most efficient way
    - Need to use SBOMs to make decisions

Vendors have vulnerabilities that they do not address

- Case for transparency – can't make the vendor do anything, but it there any remediation that you can do in your systems security brand that they might not be able to fix

On prem – service vs. cost (less applicable to particular case)

- Terminology issue based on use case

Virtual device vs. virtual machine

- Device – their software running in my cloud

When buying a service, becomes iffier

In device case, SBOM matters differently than in other cases

Recommendations for vendors that don't update – get another vendor

- Based on the expectation that vendors will investigate and remediate

Cloud and services have much more rigid contracts

- Struggling because accountability and technical?

Independent of the BOM, there may be value in hooking other types of communication to SBOM to explain vulnerabilities

- Communicate issues about the false positive vs. "naïve trip"

Do we want to have a conversation about what pointers may be?

- Cross-reference databases for potential vulnerabilities?
- ➔ Do we want to enable further communication for the vendor to the consumer?

Three columns: Column 2 is the most dangerous

"this is not vulnerable in my implementation"

- Need to push through straight to the 3rd column

10/100 are exploitable

- Put in a column as the supplier: threat level x, not implementable at this time

Log4j vulnerability

- <5% vulnerable, most don't use the feature that was vulnerable

Need to address:

1) Naming issue
2) Vendors need a way to say "this vulnerability is not exploitable in this product"

- ➔ way to put online
- ➔ people applying fixes that don't improve functionality – can't handle so many fixes

To what extent should we be tracking SBOMs?

- Has to be independent of the BOM
- Just communicate our mitigation

"Nutrition label"

- If pushing int build time, need to include all things that are potentially in component
    - Not digestible for customer

SBOM: add another column?

- SBOM = ingredient list
- Need something else to relate SBOMs to cde

Need something to prove that 3 months ago, you were compliant

SBOM different than assurance package, but sometimes lumped together

ACTION ITEM: what word refers to what context – framing

- Nutrition label vs ingredients list

2. Convey data

    - clarity on versioning

    - naming clarity (standards and formats and framing?)

3. Use-case (cloud)

    - need to also look at differences in contractual accountability

Useful vs. not under our control?

- Package up and leave it unresolved?

4. Vulnerability and communication

Next steps:

1. In-person meeting time period of Feb 20-27
    a. Happy to go on the road, worried about timeframe during RSA
    b. Conflicts with European conference
    c. Feb 11-15 HIMSS in Orlando

2. Virtual meeting 2nd-3rd week of January?
    a. FDA workshop: Jan 29-30
    b. Will circulate something a few weeks after the holidays
    c. Create intermediate draft

SIGN UP FOR WORKING GROUPS

- ■ NTIA website (October 1st update)
- ■ Linux foundation
- ➔ Information circulated in the next day or two

Who else should be represented?

Currently represented:

- ■ Look at guidance doc for sector-specific participation