This document was compiled in real time from the multistakeholder discussion at the July 19 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

Use Case Discussion

Concern: 100s/1000s products, customers around the world, how do we, assuming info about components and vulns, how do we provide info to good guys/customers without providing it to the bad guys?

The notion that the bad guys don't already know is crazy. The defenders are the last to know. Assume everything is already compromised.

Use case: Defenders should have same info that attackers already have.

Most CVEs are found from the attacker perspective by individual researchers.

Starting with a smaller and more easily contained use case with proof-of-concept testing may be the way to go. ID something that works in health care, automotive, or critical infra, see how that works, then look at expanding more broadly.

Many components are general purpose, so smaller scope would still require involving a significant number of components.

Small scale is best. In health care, several medical manufacturers/hospitals are interested in participating. Treat millions of patients.

Leverage industry groups with smaller number of participants.

Must be scalable, machine readability, consistent naming conventions

Use case: look at cases where actual exploits occurred, and look at what could have been done with transparency to mitigate them.

What are the outcomes we want, and how do we work back from that.

This has been done at scale in financial services for 6 years. Their processes are mature. Deployment modalities change.

Use case: Know where vulns are and where they aren't.

Open data: info is sometimes encumbered by intellectual property restrictions – the commons for sharing info must encompass the right to share info that is unencumbered by legal restrictions. (Open source software today, vendors have requirements)

Should we think about the use of the product, eg is it public facing or more restricted?

How can NVD encompass more vulnerabilities?

Use case: Where is the waterline for what I don't have to tell you? We need a single standard. ("spices")

What is our scope? Probably not SBOM w code snippets. Third party components? Need a scope that is workable across sectors.

The CVE accounting issue is different from the SBOM. Can still get started on the SBOM.

Device manufacturers required to say what platforms they are on, but it's not required to align with vulns. But who should be seeing the vuln info? Enterprises or end customer?

When a new vuln is disclosed in a component, if you have an SBOM, all products that use that component, it's public that it's disclosed. There's not necessarily a separate disclosure.

Focus on concern about exploitable vulns. This may already be tracked, and could be shared with consumers in the context of sharing vulns that are not exploitable as well.

If we tried to identify the characteristics of things we need to look out for – we would be fighting the last war.

Exploitability index at MSFT – idea was to predict which vulns were most likely to have reliable exploit code avail in 2 wks. Was hard to educate consumers on how to use them. Even if we produce it, we have evidence that consumers don't use the info, they just go by the criticality overall.

Vendors make mistakes, they may think we won't be affected by an exploit but we are. Assumptions may not be accurate. Hospitals want to play a role in our security posture and we want to work with suppliers on this. Could envision scenario where we look at how info is conveyed, how customer leverages it, feedback mechs, lessons for other verticals.

Nutrition labels are useful for people who need to know what's in there, eg health issue or allergy. Puts risk decision on operator.

For most open source and 3rd party components, an update includes multiple fixes. We aren't suggesting that consumers go and point fix specific CVEs. We are talking about component update that fixes multiple CVEs, some of which may be exploitable.

Use case: Business process issues behind transparency – eg having un-updateable components was declared to be bad. Is something automatable? Are things machine readable? The resilience issue is more important than the vulnerability issue. Business processes allow you to address the vuln.

When you can easily produce an SBOM, we know the team is highly efficient. The processes that allow them to create an SBOM are highly efficient. (SBOM is a side effect)

Learn from existing best practices etc.

Would we criticize the govt if they were not transparent about a new flu variant? Yes, we would want to have the chance to decide for ourselves. But we want enough people to get the flu shot to avoid collateral effects.

Core issues: we want people to be efficient and productive. Helps with security too.

Responsibility for those of us who write code for public use, if we know there are vulns that may be exploitable or not, they may be exploitable tomorrow. We should aim for a single standard.

Challenge is the large number of legacy devices – no way to upgrade them. (big issue in health care)

Use case: make sure that whatever we put in place can safely be used with products that can't be updated.

SBOM can be retroactive, enumerate the old stuff.

Financial sector – what has their experience been in terms of using the SBOM? Responding to vulns, taking action to use info.

Based on an FS-ISAC paper from 2015, includes contract language sample. Each institution decides whether to implement that. Not totally consistent. Usually new companies don't have a hard time with creating an SBOM. Legacy vendors have a harder time.

Use case: If a vendor can tell me what is in their software, I make a certain risk decision based on that knowledge. If they cannot tell me, then it becomes a negotiation point for me. I have to spend more time and $ to determine what is in my environment in the case of an event – costs more downstream. My contract negotiations then include concessions because of that.

Large bank spent 3 months with 4 people going around the org inventorying where they might be affected by particular vuln. Other entities already knew instantly whether they were affected. Also many customers approve software to improve internal practices and also to analyze things they are buying from other people.

Ask orgs unit of time it would take to know whether a vuln is in their system.

Vendor stating they have an SBOM gets us a long way toward secure products. Would be even better if they had a process for dealing with vulns. Should SBOM be public?

Use case: Distinguish between software product and devices (more closed and harder to update).

What are we trying to target with an SBOM? How close does SWID get?

SWID allows enterprise to count the number of software units they are using to appropriately pay vendors. But most open source developers don't know what SWID is. Because it's designed to get people paid, there is a relevance shortfall. Many commercial vendors do use SWID, and there are efforts to increase use across open source.

SWID vs SPDX – interoperable?

Issues with version numbers, old software, necessary info far too granular for SWID. Eg shellshock vuln.

Possible workstream: capture existing observable practices and IDs most effective ones.

Open source developers are unlikely to adopt establishment solutions, but most open source products are built on native systems. PURL attempting to allow native systems to interoperate under one umbrella.

SBOM = labeling every package of ground beef with the cows. Highly complex.

Scope and scale – must define carefully.

If you aren't automating this stuff, you're already in trouble. Transition from artisanal to industrial process.

New thoughts after the lunch break?

Process and format in the end result are the same thing. Standardizing processes requires automation, scalability.

Vendors are stakeholders in the success and protection of our customers and communities.

There's a difference bt products and platforms. Products = solutions. Platforms = enables set of solutions that enables and ecosystem of solutions to be delivered by others. SBOM enables safety so it's a platform. Empowers those with the software components to become safe. Risks can be addressed through compensating controls.

[Discussion on defining what we mean by transparency using Jim Jacobson's slides of three dimensions of depth (what is a component) breadth (how much detail about component) and time (how often to update the list of components).]

SBOM Depth – how many items are listed (recursion, snippets, firmware/hardware, "all" software components, attack surface-relevant components)

SBOM Breadth – how much detail per item (component name/ID, major/minor version, patch level, attributes/context)

SBOM Time – how often to update (at delivery, major update, any update, when patched, in real time)

Our effort should probably include third-party external components to vendors' products.

Breadth – dependencies that are not components is an issue

When should SBOM be updated? Seems like whenever a patch is applied or any versions change. Not in real time but it should be an accurate reflection.

Why is this not a standard part of releasing an update? If they were machine readable then they would be much more useful. (language is also key.)

Could create a best practice that release notes should be machine readable. (out of scope?)

Limit to on-premise or not. If you're going to do cloud, you should just be able to have an API to ask for inventory.

But most devices talk to the cloud, so how can you ignore that?

Thinking about clouds might help us think of SBOMs as an automated thing as part of the process.

SBOM spec would need to be simple

Keep focus very narrow.

Not a big fan of “SBOM” – there is vulnerable hardware.

Whatever we come up with should be lifecycle agnostic.

Focus on on-prem software as well iot devices

The minimum expectation: What should an SBOM include: All software components, component name with major and minor version, with each update.

[back to main pc]

Focus on on-prem devices and some devices? This would help clarify why you want to do this – receiving something from someone else, you want to know what’s in it.

You need to know who would find the info useful before starting to pull it together.

I want to know that that Saas provider cares about security, so there are many reasons to request it

Deeper issue: supplier, deployer, securer, consumer – should the info be exposed to those who consume it? Much more complicated. (Not all agree)

How much benefit is there to telling customers about specific issue they can’t remediate? Meanwhile, a car manufacturer needs to know about issues in entertainment system, but the driver doesn’t.

Supply chain is compelling argument. SBOM can push vendors to level up security maturity, we are using hammer vs needing screwdriver. Other tools to apply.

Some users are able to use the information better than others.

Sometimes the user is the securer *and* consumer. SBOM is probably most targeted at the securer.

Usage scenarios are critical piece to understanding what can be done.

Developer/producer should have command of their 3rd party components as part of building a secure product. Getting that confidence may not require releasing a BOM as long as the developer has it and is using it to stay on top of what is happening with his components.

Does an SBOM need to be released/public?

Start small.

Use existing tools. Don’t want to leverage a lot of heavyweight processes on vendors bc they prob won’t adopt them. We do want to look at how vendors could integrate this into their build process. Then it could come out as an artifact of a normal build environment. Then the software producer could decide if they wanted to ship it or not. Machine readable. We use SWID today. Other large vendors do as well (and some small ones). What would it take to incorporate component info – not a lot.

Must be reasonably adoptable, deployable, and lightweight.

SWID, SPDX, open source prefers SPDX, open source spec CycloneDX for a lightweight BOM, supports applications and hardware, can be integrated into a build environment. Open source commty is open to this. Package URL spec could be really useful (CycloneDX includes this as part of the data spec, uses those ecosystem's native package mgrs to standardize what the resulting URL could be).

Every build system is already using very specific coordinates for how they manage their dependencies. Whatever coordinate system we use needs to be precise enough to carry the fundamental info

As an asset manager, if I could get a basic tool like an OS version and a ? number that would help a lot.

Start with something super basic.

Device vendors – much of the necessary info for a device doesn't come out of the build process, it comes out of the platform.

CoSWID – being developed in IETF now. Makes all the attributes textual indices, makes SWID file much smaller for resource constrained devices.

MUD anomaly detection, draft standard

Could the MUD model work in this context?

SPDX – package URL spec is going into SPDX in next version. Looking at interoperability with SWID. (Kate from Linux Foundation can provide more info.) Being used in Europe, cleaning up Linux kernel, captures CPEs, can correlate with SWID too. Specs for SWID reside behind a paywall.

We see tools doing name-based matching, but you have to be precise about what you're using. Many items use the same name.

Whatever level of complexity we choose, we must appreciate that just knowing what the component is is not an indicator of vulnerability.

FORMAT vs NAMESPACE – SPDX is the format, PURL is the namespace.

Some languages will be more challenging.

Vendors track software composition analysis. Waiting for a standard to emerge

Working group possibility: Using SWID and SPDX – what does that look like for SCT?

Challenges

Metadata? Guidance? How to address challenges?

When a product reaches end of life, you'd like it to have an SBOM

Working group possibility: How do software producers maintain list of residuals and dispositions across stakeholders and time?

Backporting, now unsupported, not upgraded to new version but can apply patch. There's no standard format for that.

Tracking mitigations – format, complexity, depth of analysis. Eg, linux distro where packages had vuln, vendor said we will update this package but you are not vuln .. bc this component is not turned on by default. Set of contingencies vs yes vulnerable or no vulnerable.

Lack of demand signal in the marketplace – got "un"usual suspects to go to vendor booths and ask for things.

Not all CIOs and CISOs know they can even ask for this. Adoption and awareness piece. Coaching on how to use it.

House E&C had roundtable with medical CISOs. First they thought it was too much info when it was on a PDF but they could use it in other formats. Machine readable. Road show across ISACs.

Whatever comes out must be vendor agnostic.

Working group possibility: Medical device pilot/experiment, including issues of publication, consumer consumption, sandbox. Peer institutions as consumers, we need med device manufacturers to try this with us, find some commonly deployed devices, pick a standard (or 2) and have a whiteboarding session where we plan what this looks like. Can this be informed by other actions from this MSH? (or vice versa?) Health care is not different – the solutions are very common. The current status quo doesn't scale. Wannacry still an issue bc vendors aren't managing patches.

Also look at integrators and deployers, and time and effort from each participant. How doable/repeatable is this practically, and what is the resource requirement?

Also get info from financial services, they measured cost savings. Let's measure it all. (Awareness and adoption – document lessons learned from those who have done this.)

The term pilot is dangerous. Mistakes get baked into next steps. Encourage using different term like "reference implementation" so it doesn't get baked in. Or "experiment."

Other potential issues: Vulnerability vs exploitability; Will tell most but not all; FAQ (to answer objections);

Defined scope/goals are key. Need to know what you're trying to achieve to see which objections to focus on.

Survey of intended consumers re what their current costs are for deploying patches. Frequency of updating information – organizations struggle to handle the vulns they already know about. What they spend now, what would they like to see? Also analysis about what the cost reduction might be in improved efficiency from having the info.

Risks of not doing this properly – how many orgs have efficient, fully vetted patching processes.

Metrics for success – benefit of adoption could be emergent and distributed across the whole ecosystem vs being accrued by single org.

End user-operator can choose to ignore info, at least they have it.

Over 10000 orgs downloading known vulnerable versions of struts. As we work to make this more of an acceptable best practice the side effect is that everyone starts paying attention.

Intrusion detection systems don't work the way you think. They provide visibility into the network, forensic data after event. One reason is false positives. Signature is accurate 99% of the time but .1 of a billion packets is still a million packets that trigger events. Can I give you an SBOM that you could map to a vuln database that results in more signal than noise? There will be lots of false positives.

Need a threat model that connects those people together.

If the SBOM has info from the vendor saying it's exploitable then timeliness matters. If you have 300 open source components you'll get a lot of CVE hits. If the vendor updates you monthly or quarterly there's a long delay before notification. If SBOM can contain metadata, SBOM may be updated more frequently than the product itself (critical infra, medical world).

Updates must be continual.

We don't want the vendors to have the same info. We want them to have the same understanding. The vendors that don't end up taking random actions. And usually that's nothing.

Stop the paternalism. Business process – shift from batch to continuous. Continuity + automation.

If you know the info and don't provide it, at what point does this become a legal/negligence issue? Why not disclose?

You have to be willing to ship garbage to have the conversation about what users actually want. Let them make mistakes, blame you for the mistakes, learn from each other. We will have to teach people what to expect and what to do or not do with the info.

Completely clean SBOM is unrealistic in most cases.

Hospitals are critical infrastructure. Security used to mean HIPAA, was laptop encrypted. Now is much broader. Working with LE on ttx blended event that brings down elevators, can't move patients, can't deliver care.

A lot of the concern is how the recipient interprets/uses info. This could be a use case.

What can we do today, and what does transparency look like tomorrow?

Are we talking about warnings as well? If we are talking about having data available, what's the incentive structure for building a common knowledge around that?

Understanding vs raw info: We have 30 years of experience with full disclosure. Or do we try to give understanding. Some vendors may corrupt their data for marketing reasons. Other vendors see it as an integrity issue and disclose all. Interests are not aligned. We get the raw data out there first and understanding comes second. Understanding may come from other sources.

Also concern with over disclosing bc it's not that useful. Lead paint disclosure example

How can we measure the benefits in a way that helps us to understand what costs we'd be willing to bear? Examples?