>>: We, um, are looking forward to a productive afternoon. If you are just joining us here in Washington, DC or online, um, we had a productive morning, we heard from the four working groups. The further materials that were presented are going to be available on the NTIA website, and I'll send out a note, once we have everything collected. We are also going to be posting notes from this discussion, and in a few days, we will have, in fact, the entire video stream, so you can re-live the highs and lows of the experience today. Um, what I wanted to do this afternoon is a few things. First, tackle some of the questions that we've heard raised, um, over today's meeting and over the course of some of the past working group calls, I want to make sure that we've got a full list, so if there's anything that we don't have up here, I want to make sure that we can put it on this sort of new ad hoc agenda, and then, finally, before the end of the day, I want to sort of revisit the process. Every NTIA multi-stakeholder process is a little different, one

of the things that makes this quite unique is we are all approaching a fairly similar problem, but from different perspectives, so there's, in the past, the different working groups have been a little more orthogonal, a little more independent. Here, there's a lot more interdependency, so I want to talk through how the different working groups are working, and then, finally, we're going to talk about where and when we will meet in late February, and if we want to have sort of an intermediate virtual meeting, um, in mid-January, just to give us an excuse to do some work over the holidays. I know that you love it when I frame it like that.

So, um, these were some of the questions that I flagged and tried to sort of outline before the lunch break, which is, you know, what are the components of a software bill of materials, and we may drag Art up here and Michelle up here to sort of talk about what they've captured so far. Um, how the data is going to be shared, so, you know, what is the act of transparency look like, if you like the framing of software transparency, how do we make sure that it's protected, but still effective, where do we store it, can we at least, we don't have a single way to do it, can we make it a fairly small list, again, focusing on harmonizing it, making sure there's a relatively well-trod path that most vendors are going to follow and most customers can expect. Third thing I want to talk about is something that Omar flagged and a few other people flagged, which is, hey, how does this translate into the cloud, right? It's 2018, and we're still saying, but what about cloud? In this case, I think there's really some value in understanding what makes this very similar, and what, if anything, um, is different, so that we should understand how to apply it, and then, finally, I want to come up with, come back to a question which has, um, was at the first meeting, and it's going to be something that we're going to raise for awhile, which is software bill of materials may be very useful, but we may need a little other related data as well, and we don't have to define what that is, but we can at least start to talk about it, start to talk about how it'll be pointed to, start to talk about what makes this data, um, as useful as possible to both vendors and customers, and so what else do we need, what does an extra data plan for something like that look like. So, is there anything that you would like to add to this list of open questions? And if there's anything that you say this topic is verbose, and I think it's a waste of all of our time to talk about it? Josh?

>>: I think the going beyond column one from the graphic I pushed around is both necessary, well, three things, necessary, valuable, and rot with peril. So, what I'm worried about is I really, personally, want to figure out how to make those nutrition labels, and myself as a producer of software, in a sane and consistent way with the rest of industry, um, but I'm afraid if we do that, we're going to have to put up some ground rules, because it might get messy or ugly, because then we're going to trod into the very territory we've been tip-toeing around thus far. So, I don't want to avoid it, but I want to have some constraints on it, I guess.

>>: All right. Thanks. Bruce, is your light on?

>>: Yeah. Um, this might be communicating related data, but if it is, I don't think it's clear from those words. Um, there's requirements for this to be successful that, at least when we've talked about it in framing, we've said isn't part of what we're doing, and one of them is naming. Naming is a huge problem, and it could be handled independently of this,

but someone needs to do it, and the naming issue is basically, I'll call it aliases, there's 18 names for the same thing, okay, we need to know one of the same thing. Everybody, I mean, all the large manufacturers have acquired lots of companies, and every time you acquire something, there's a good chance that the names are going to change, at least the name of the vendor, and, um, you know, in order to know that, what you're dealing with, this has to be handled.

>>: Great. That's a good thing to make sure that we get a chance to touch on. Anything else in the agenda whacking part of the discussion? I think this is certainly enough to keep us busy. Um, so, again, for those of you who are watching or listening on the phone, dial star 1, and that will put you in the queue, and we will bring you into the conversation here in the room. Um, so, if we're happy with this, we can dive in. You want to start with the question of components? Art?yes?

>>: There are probably a lot more questions that go up there. Is that what you're asking for?

>>: It was more what are the open questions we want to tackle now, but if there are, I know you have a whole list in your document.

>>: Yeah, I was trying to match them up. They might overlap some. I'm good. They'll come up, if they come up. So, thank you.

>>: Great. All right, so, we can, um, start with something that, since you had the mic last, um, what are the components, and, you know, is this something that we want to sort of imagine a phased build-out of? And is there sort of the, let's focus on what it takes to get the job done. Um, can you give us a reminder of sort of how you guys managed to narrow it down so that you weren't talking about everything? Michelle, yes?

>>: Sorry, I just want to ask a clarifying question, and not to be nit-picky, but components as we've been talking about them mean actually the software, um, so, if we could, maybe, put the, I think one of the things that we've used is SBoM elements or something like that, um, just for consistency of language, the components just means something else, I think.

>>: That is very good. Thank you. I appreciate that. So, I think in the discussions that you've had, and it's also come up in some of the other groups, you've narrowed it down from a fairly large

spectrum to, um, a number of finite options. Do you want to sort of put those on the board? Or is that not a fair characterization?

>>: Um, I think it's a fair, let's take a shot, I think. So, um, this is Art talking. So, elements of an SBoM, we just had a discussion about naming, but we need, I'm not sure naming can be outside, there needs to be, um, our minimum viable product had a triple string of, um, vendor, product name, version, wide open strings even, which is not a real sound naming thing. These people have better naming and ID, but I don't see that being outside of the SBoM, or else I don't know how you talk about the parts, components, sorry.

>>: So, the approach that we took, and it's not necessarily a universal prescription, but we wanted to be back-compatible, was to look at the CPE naming conventions and to be able to use whatever the CPE was, if there was one. If there isn't one, which is often the case, um, we essentially alias a name that is congruent to the CPE naming convention, so that if and when a CVE emerges, it's very likely to be identified with that name, and that was a kind of practical approach that anticipating a customer, essentially anticipating a vulnerability-checking, and that's just

one of the things that you can do.

>>: Sure, does SPDX or SWID have a comment here?

>>: So, the other thing that can be, the other one that's, um, out there is looking at the repo that the source is coming from, and, um, so the pearl specification was mentioned last time, package URL, and that is a way of identifying a component uniquely as well

>>: And I noticed that we have, we're lucky, you mentioned CPE, we have the driver of CPE, Dave Waltermier, here with us on the phone. Dave, can you chime in?

>>: With this identification conversation, that there's, um, there's a timeliness problem when it comes to, um, to software identification, and this is one that's really evident, um, from a CPE perspective. As some of you may know, we run the national vulnerability database, where we try to map, um, vulnerability to vulnerable products, and one of the challenges that we have is, um, identifiers for software, CPEs in this case, are generated by our analysts in the NVD, and this happens after a product has already been fielded, um, after a vulnerability has already been discovered in the product, and our

analysts then have to, you know, forensically discover information in order to try to come up with a name, you know, for a product. Um, that's way too late in the software life cycle to actually have a reliable identifier for the software that we're trying to, um, to identify. I think where this effort could really help is by, um, accelerating the identification of software, so that, um, an identifier is available when the software is released, before the software is installed, and have that be a persistent identifier that can then be mapped to, you know, by a number of different disciplines, you know, vulnerability management, you know, being one of those disciplines. Um, CPE is just really inadequate from that perspective, we need a better solution.

From this perspective, we've been looking towards using vendor-provided, um, software information, um, as part of a longer-term solution, because that's the only way to assure that, um, you know, property identification information is provided, um, at the point of, you know, distribution of that software, which is where it's really needed, um, you know, and the formats that we're looking at, you know, around, you know, SWID tags and SPDX, um, have a potential to address this problem. Now, at the vendor source, the problem that I think we're going to run into is if there are multiple software identifiers for a software product, um, then we have, um, sort of a multiple mapping problem, um, which is also then going to complicate all of the use cases that require some mapping to software. So, ideally, if there was a single installed software identifier, um, that's something that we could map to, um, from the vulnerability use case. That would certainly simplify things dramatically. So, I just wanted to throw that in.

>>: Thank you. I know you've been thinking about this for a long time. I've got Josh and then Kate.

>>: I feel like we got into CPE and SWID and SPDX, maybe way too fast. I mean, the bare bones are library name and a library version, and, you know, maybe for bonus, if you want some sort of unique checksum that could replace the version, if you don't have versioning for more primitive language, but it's basically, like, what's this thing I'm pulling into my build, or putting into my software, and which version of it, because vulnerabilities map to versions, not to projects,

right? Licenses map to projects, which is why that was acceptable, but vulnerabilities tend to map to one or more. Um, as far as the plural names thing, you know, in theory, if we wanted to future-proof this or make it more robust, you could have an aliases, you know, kind of field or array of some sort. Um, I think when we get to the cloud stuff, just because I had the mic on, um, some of these things need to be looked up often, in which case there might be a resource to interrogate, the address of a resource to interrogate, that could be something like the pearl stuff I haven't looked at yet, um, but if we're going to try to suture this, and the room doesn't know this, but if we're going to try to suture this to the software creation birth moment, the as-built, then we wouldn't want to do this after the fact or have NVD assign something later on. So, the temporal problems come in when we de-couple software creation from software tagging, right? This is your born on bar code, you know.

>>: So, to pick up on that, the element of the time and when you build it, um, the thing is, you have multiple people building it, and all of these things may have the same vulnerability, because they're still working from the same source, and, so, while you have

one building it one way, another building it another way, and another building it a third, they're all belonging to the same upstream, so they still need to be sort of connected up in some ways rather than just who's building it, it needs to basically still probably go back to the source when the source is available.

>>: And I almost, I also cringe when we say we have to know the company name. To Bruce's point, companies get bought and sold all the time or go out of business. That might be a useful historical context, but I'm not sure it's the most actionable bit for our use cases.

>>: Okay.

>>: Even names of products had a different name in the past. I don't know if anyone knows what the name of it is here, but I bet not anymore than two or three. Everyone thinks of struts II, but the names of products change all the time, it's not just the vendor.

>>: So, this is, there's a temporal aspect of names. Kate, you seem to imply that name is insufficient, we actually need some more context data?

>>: I think if you're looking at what you're doing at a point in time, I think having the hash associated with what you're talking about is always

going to be key at this point now, but having the name and the version information is a starting point for giving you an indicator to know if you're talking about the same thing, but the hash that one person's going to be build is going to be different than a hash another person is going to build, but we're still talking the same source, probably still the same vulnerability that could come into play, just depending, like, say, if it's a slightly different version of the compiler, it might come up with a different binary. That's all.

>>: Omar, then Art.

>>: Yeah, so, as a consumer, from a vendor perspective, I agree in the naming convention entirely, and, of course, the dilemma of acquiring companies, that's a different thing, but what I wanted to point out, I like the simplicity of that. The challenge that I'm facing, at least from my side, even if I say, okay, this is product X, Acme product version one, two, three, it's also running open SSL, or I don't know what they use, and several components, so for me, whenever I publish that bill of material, it's only product Acme one, two, three, versus everything else that is three or four layers down the stream.

>>: Josh?

>>: Um, just a clarification on that, this has not come up yet, we've been debating other points, but some folks have said is this a flat list, or is it hierarchical. I, in most of the policy debates I've had over the last six years on this, you are, might make Acme one, two, three, bottle of soda, but you'd be responsible for everything that came in before it, and in many cases, you can't even build it, if you didn't have that stuff, but, um, that's why this, that's why we've oriented our spreadsheet as a relay race, or as a daisy chain, because if we don't have a consistent way to communicate downstream dependencies, we'll never get that line of sight. So, my expectation isn't that we're just going to say if you're a Siemens medical device, you're going to have just the pieces you took, we're going to have to list the pieces you took, plus the pieces I or someone else gave you, right? So, it's going to be the superset, and one of the ways is to show a small post-it incatenated to other small post-its, which become your master BoM.

>>: I've got Art and then Dave.

>>: Quickly, just to go for a moment, so I'm very happy with the tuple or the triple, I like vendor name, product name, perhaps the minimum requirement option

to have hashes or other things, the reason I bring up existing other specs, for instance, an SPDX, when you think about it for more than 5 minutes, naming is harder than let everybody make-up three strings and see how well that works out. It may be functional and useful, and that's great, and it's a fine place to start. We already have in the room knowledge about next level of what you can get to, so we should at least sort of consider that. Um, and, again, we were talking about minimum viable, so maybe the hash is, and the more, um, complex or complicated naming is, um, is at a later step.

>>: So, um, Omar talked about sub-components. Is there anyone here who thinks that we are talking about something other than a recursive list? That is when you, yes?

>>: Someone on the flat, um, a flat Earther should chime in, a flat SBoM person, because I don't get that argument well, and I think we should hear that argument.

(Laughing.)

>>: I mean --

>>: You're associating the flat Earth people with the flat, um, SBoM people.

(Laughing.)

>>: My understanding is the flat SBoM is I've got a product that I've acquired somehow, and I have the sheets of paper that have everything in it, they're all right there. Actually, I don't really understand the flat one very well. The hierarchical, relational one, I think is the right way to go, but --

>>: Why, I mean, in the real business world, the question, the question, is what is the technical maturity of the customer, and, so, you can create some wonderful, beautiful things, but if the customer's technical maturity is anything with a CV of a 5 or below goes, right, and you can very cheaply generate a flat SBoM that's going to satisfy their security criteria, that's what the market will do, and, so, I think that there are elite customers, right, so I would put, like, a lot of financial sector customers in this bin that are going to want to do a ton of really fancy stuff, um, and, you know, companies help them do it, we help them do it, but it is actually shocking, like, how low the bar really is, and, so, the strategic question is would you rather get more adoption of an easier and flatter thing first and evolve, or wait and come up with tooling to create, you know, these sort of system

descriptions that more sophisticated customers are going to make better use of.

>>: So, I know at the last meeting, we did have some discussion about should we aim for more sophisticated or less sophisticated information. I want to give sort of just two or three very, very quick responses to this, and then I want to keep going, because we had some folks on the call. I've got Jim, I've got Josh, and I've got Mike.

>>: I think you have to start unsophisticated, because in order to start at the very highest point, you have to have all of your suppliers having their SBoM provided to you in order for you to incorporate that into, or to have it available in some format, and, so, there's a chicken and egg problem there.

>>: Um, Josh?

>>: Not placing value, just to describe, I showed a slide that had a Cisco SBoM, it was flat. We could show the output at every single commercial tool, they start flat. If it's additional context for people that want to do variant analysis to see what else might also have this CVE that hasn't been pulled in yet, there's definitely additional value there, but the current state of practice is flat.

>>: I want to distinguish, I think there's flat, which is to say there's no information, and then there's non-recursive, which means I only give you the top level of what I am shipping and don't give you --

>>: The dependencies of dependencies show up at a singular flat format. They are comprehensive.

>>: It is recursive, but it isn't semantic.

>>: Recursive without semantic value. So, let's, and Mike, I want to get to your point on the maturity of the customer, and then I want to sort of pop up to this level.

>>: I was just going to mention that I think there's, I'm not sure what you were specifically referring to, Josh, but there's a difference between the stored version and the display version. You can always flatten a hierarchical BoM, you cannot unflatten a flattened BoM, you cannot make something like that unhierarchical.

>>: Yeah, a lot of these tools do --

>>: So --

>>: They don't show it that way.

>>: Just to finish the thought, don't foreclose hierarchical.

>>: Okay. Um, so, I've got three people

waiting on the phone, but I want to give one more chance, um, because, Jim, it sounded like you were saying only top-level includes, not recursive.

>>: I was saying that initially, that should be the target.

>>: I think we need to clarify what flat means. It either means just the top level, or it means everything, no matter what level it's at, and those are two different ways to think of flat.

>>: I think we're talking about all the transitive dependencies, but no semantic relationship between them.

>>: Well, some people are, and some people aren't, so we should define it.

>>: Does everyone like JC's definition, which is, um, we've got, well, no, because we can still say, right, there are different, so, flat, JC, can you repeat this?

>>: So, all transitive dependencies with no semantic relationships between them.

>>: And can we get a term for if you were only describing the top-level includes, what would we call that?

>>: Direct dependencies.

>>: Okay. So, we have direct dependencies, which is what the data that you're including is, and then we've got flat verse hierarchical, which is how you're given the data. Does everyone feel like that's a good way of talking about this right now? All right. We're going to the phones. I've got David, then John, then Bill.

>>: So, I was thinking about this transitive dependency issue, and I'm not certain I actually see a significant amount of value in capturing transitive dependencies, if there isn't a semantic relationship between the dependencies, but I think in general, what we want is a model that degrades well. So, we want a model that can support, um, a flat representation, when, you know, when information about those dependencies, um, is not available or can't be published for various reasons, and we want a model that is capable of expressing, um, the graph of relationships, um, as an organization can provide that kind of information, and I don't think that's impossible, given the current state of things, because, you know, both SWID tags and SPDX already provide a fairly rich semantic model, um, for expressing, you know, that kind of information. Um, so, I don't want,

I think this effort shouldn't sell itself short, I think we could provide something that is immediately useful from a flat perspective, while still providing room for growing with a much more robust, um, you know, transitive dependency model. I wanted to go also back to the identification question.

Um, I think there are a bunch of different use cases in which there are different identification requirements. Um, one way of looking at identifying software from a vulnerability perspective is it's not enough to actually have a tuple, one that would say the vendor of the product, the name of the product, and the version, we really need to start getting down to, um, the sort of package and release level identification in order for that to be useful, because there's all kinds of magic that actually happens behind the scenes when software is being released. Um, some will add functional patches to software, other will, um, will add, um, will back-port security fixes into software, so, um, it's not just that you're getting a unique build of the software, you're getting, often, a unique version of the software that doesn't directly align, um, to, um, the, you know, open source version.

>>: Dave, you have, um, strong enthusiasm in

here from Lennox Foundation's Kate Stuart, so we've got at least some support here.

>>: And, so, I think, you know, we started before the break talking about all of the various, you know, use cases in which we need some SBoM-like thing, and someone even brought up the idea of, you know, do we still call this an SBoM. I think one of the challenges that we have is sort of a lack of vocabulary to talk about the various formats that we need to satisfy various use cases. We're calling everything an SBoM, but what we're really talking about are really a handful of different things that are needed to support different use cases. We need, um, we need meta data about installed software, we need providence information about how, um, you know, software is assembled and built, you know, that includes license information, and these all satisfy different use cases, different stakeholder groups. Maybe we should be talking a little bit about, um, what those different stakeholder groups are and what information they actually need, and maybe start talking more specific about the specific data bundles that are needed to support those use cases. Just a thought.

>>: Excellent. I'm looking over at the chairs

of the use cases working group, they don't have to comment now, I'm just, we can talk about having that, folding that into their role.

>>: I'm sorry, that was a bunch. I didn't have a chance to jump in earlier.

(Laughing.)

>>: We are, the phone model has some issues. We have a quick --

>>: I think that, I mean, there are customers who really do care about chain of custody, so kind of where the thing came from, whether it's a package URL or, like, you know, this was the time-stamped HTTPS session that this particular tympanic came from, to some sophisticated customers, that really matters.

>>: Great. Um, we have John Willis on the phone.

>>: Yes. Can you hear me?

>>: We can. Thanks.

>>: Okay. Um, just wanted to kind of go back and talk about two basic configuration management type terms, one of which you've already addressed quite a bit since I pushed the button, regarding the transitive dependency. So, the terms are basically what is this software used by, and what does it use. So, the

transitive dependencies would be what does it use, but what is it used on or used by is another concept. I think that that one in particular doesn't, um, relate in most contexts, but in some contexts, it does.

>>: Thank you. We have, um, Bill on the phone. Bill? Un-mute, if you are muted. Bill, we're going to give you just a little bit more time. We gotcha.

>>: I have a mute on my phone and speakerphone, so go figure. So, two comments. One, from a naming convention perspective, typically, names are made-up in marketing, not by anybody technical, so I would place not a whole lot of faith and ability to track things, um, definitively by product name and version, because they'll change, right? They'll go from, you know, product name, version two to some other product name, version one, when, in fact, it's the same product with a few minor changes, so it gets very interesting very quickly. From a flat versus hierarchical versus, um, transitive perspective, um, you know, my perspective is from a manufacturer, um, I don't necessarily care, again, how things are plumed together, as long as I have a concise list of what things are in a product. I don't necessarily care for the purposes for what I need this for, that's what the hierarchy is. Others will, right?

>>: So, Bill, I'm going to push on for two questions. First, your second point, do you care about the transitive side of things? You want it transitive, but you don't need the semantic data, is that correct?

>>: I would say that's correct, yes.

>>: Okay, and the second is you mentioned that names are really hard, because they're put forward by marketing people, and can you believe those guys? Do you have a solution or a path forward?

>>: Well, I mean, it's what you typically see with a lot of other things, is you come up with some lower order, um, meaning convention that is only tied to the actual product name in some other way, right? Um, whether it's a catalog number or a skew, product, or what have you, it does have to be something that's consistent, but you really can't use product names, because they're never going to hash.

>>: All right.

>>: It's a new data element that manufacturers, I think would have to inject into their products.

>>: Thanks. Can you introduce yourself?

>>: Yeah. Lev, Cast Software. Just a question about the hierarchical versus flat. If you assume that every piece of software has a BoM, um, then

wouldn't you always have a flat BoM for, um, a specific piece of software, where you could actually look up the BoM of any component, and then you'd be able to kind of look at all the turtles all the way down? So, is that sort of a non-issue then, if all software components have a BoM, then you're automatically flat and hierarchical?

>>: As long as you can express a relationship between them, you should be able to do it. As long as you're able to express a relationship between these software components, a language expressing that this one relates to this one, or this is a dependency of this, or you have a way of linking them together, you're fine, you just need to have a way of linking them to add those use cases.

>>: But, I mean, if you have a unique identifier for, right, for that software, right --

>>: It has its BoM, right? You might want to link to another dependency that's potentially built into it.

>>: Okay.

>>: Or something like that. Like, if you have a container that has a variety of packages inside of it, each of those have their own SBoMs effectively, you

want something to say that these are all working together inside this container.

>>: But isn't that what your BoM is telling you, that you've got these components inside your container?

>>: Yeah, it's just a question of how do you express that set of relationships effectively, because it's not all one big document. You want all these documents that you can link together and have a master document that reflects all of them.

>>: Okay.

>>: That's how they interact. I think that's what you're going after, right?

>>: I think so. Sorry, I'm kind of new here as well.

>>: Um, I've got Josh and then Bruce.

>>: I liked your question. Um, we should also separate what might be valuable to defenders or, um, managers or master BoMs for complex regulated products from what an attacker sees, because my trial by fire into this was people didn't know they were using patchy struts, because one of their dependencies was using patchy struts. X stream is the one I showed in the real pigtails, those are really nasty and insidious, because no one knows they're using them until they're

compromised by them. So, from an attacker eye view, it is more flat, it is about column C, is nutrition label and other things, and it's almost an independent variable from a hierarchy of where the SBoM came from.

>>: Bruce?

>>: So, if you're going to say only direct includes, which I think is what you're saying, I mean, the people making the software, whether they're vendors or open source people that are putting things in, they don't really know what's in there unless, and this is particularly difficult for open source people, they get expensive, you know, scanners that go through, and as many people know here, if you get six scanners, every one of them will find something that the other five didn't.

>>: That's only if you're doing binaries. If you're pulling from source, you can use GIT to assemble a full list of transitive dependencies. It's not hard.

>>: You can sometimes, but not if I get in a library that doesn't have that information from someplace else. You know, you're talking about struts, um, being in there, you don't know it's in there unless you somehow find that out. Now, you might be able to find that out because you imagine the first day

this comes out, everybody's going to put out in SWID or SPDX format or one of the two --

>>: Or it just comes out of the build process. I mean, this is a known art, like, this isn't even, like, an advanced thing anymore, like, you can do continuous monitoring on all your transitive dependencies, like, now, it's not hard for all our open source components, because they build from each other, and they exist in repos that you can look at, and you just recursively go, like, we go through and adjust that data, and you could do it through SPDX as well. It's not hard.

>>: Well, if you want to limit this to only those components that are built that way, fine, but that's not all the components, and, um, and I don't think we should limit that, and, again, I think that the creator of the software should only be responsible for doing the top level. If they want to give additional information that's problematic, I think that's fine.

>>: I think that proposal undermines the idea of impact analysis attacks. I mean, the whole point is if you're only looking at what you grabbed, and say it's not my problem below, like, that would be, like, a car company saying I'm only responsible for the harm from my tier one suppliers, not the tier two.

>>: No, but we're, okay, so, the problem is we're not going to be looking at the source data here at all, we're going to have a tool that does this. I'm trying to figure out how to make a tool that gives us the answer, and if the tool just sees the top level and the top level, then it can provide the list --

>>: But what if every patchy project spits out something it's not giving you today, what if every project, because of this room, starts spitting out its contributory BoM, and then we are assembling and accreting and aggregating BoMs? There are some things being done today, that's what we were capturing in our user group, but what the broader group is saying is were we to do a little bit more, does this become easier? Not it is easy, but does it become easier? And if you want to be adopted as an open source project, you're going to provide a standard output SBoM that can go into Jim's device, or you won't be used by Jim's device, because he's regulated, so the regulatory forcing function causes this change reaction.

>>: I've got Art and then Kate.

>>: The point I was going to make, you guys just sort of got to. It may be that everyone looks at their one level away, if it's a network effect and that's all

in place though, it all works out pretty well. Of course, you don't have the network effect I mean, does a car manufacturer know what the tier two is doing? They're responsible, if something's wrong, but they ordered the part from tier one, tier one did whatever, right? I mean, they don't, I don't know tier eight, right? So, if everyone's responsible for one hop down, and everyone's in the network, this thing solves itself theoretically, right?

>>: So, this is, we've come to radical agreement, as long as we've got a little bit of ecosystem curation that's happening, is that fair? Kate, do you want to chime in on this?

>>: Yeah, I just think that, yeah, everyone needs to do a little bit more than they're doing today, and I think it'll help solve it for the hop-down effect, as projects do it upstream, it then ripples down into the others.

>>: And to belabor this point just a touch more, um, both from the commercial vendor perspective and from, you know, the LF perspective, is this the sort of thing where when you discover there's a package that you're using that doesn't have the right data, or when Bruce discovers it, it's a fairly straight-forward

thing, to sort of figure out how to get it in there, especially if it's in the open source world, or is that actually a fairly, a big lift for everyone to do, a little bit of situational awareness?

>>: It's still a big lift out there.

>>: So, a concrete example, so when we generate SBoM, we also generate, like, with the versions of the transitive dependencies, part of the SBoM that we kick out, um, indicates what the latest version of that component is that exists out in the world, right? So, if the component in this application is version 1.4, but there is a version of this component that is version 2.8, right, we put that in the SBoM so that you could know that there's a big delta between what the component is and the most recent version of that component, then it's up to you, whether you want to replace it or not, but if you do want to replace it, it can entail a certain amount of hand-stitching to go in, and in the build, even when you have the source, say I don't want you to use the default version that's in this build, I want you to go to this other place and pull a more recent version.

>>: That sounds good. Um, hey, I want to sort of try to capture the general situation of what I think

I heard over the last 20 minutes, which is that there is some value in having the local hierarchy data, but that might be emergent, if everyone just sort of captures the top-level dependencies, and if we have the top-level dependencies and the local structure to capture the hierarchy, we get that. Is that where we're ending up? Saying have the capacity for the first level of the hierarchy, and then rely on a slightly better, curated supply chain to get that chain of trust all the way down, is that kind of what I'm hearing? I may be missing a few things. Any concrete, am I wildly wrong? Yes?

>>: Just to follow-up, the supply chain, I think there has to be liability associated with it.

>>: A little louder.

>>: There has to be a liability associated with that. So, in other words, if, um, it's wonderful and all, but if there's not liability in the sense that if a car manufacturer is not liable for their, um, fifth-party parts, then the fourth-party isn't going to care to report out what the supply chain was for that particular party.

>>: But that's the ban components list, I think, if you can, you know, identify things you don't want,

you're essentially blacklisting components. I mean, the other thing is if you read the ULA for every single software publisher, there is no liability in software.

>>: We're going, yes, I'm always very worried about using the L word in this sort of context, but I think we can expand this to broader questions of accountability. We'll use that, accountability and responsibility. Sorry, we have two fingers in response to that point, and then we'll get to Duncan. All right, Duncan?

>>: Well, one, I do think liability is more a contractual issue between the customer, the consumer and the producer, but the point is is what we're doing capable of allowing them to write that in their contract language, are we doing something that gives that, that meets that use case is really, I think, as far as we should go from our scope viewpoint. Um, but I wanted to back up to a point that was made earlier when somebody said, um, you know, we put in our SBoM the 1.2, but we also put in that the current version is 2.4. The we in that case is this, the committee you're co-chairing, or your particular company?

(Off mic.)

>>: Okay, I was just trying to figure out if that

was now in our scope or if we made our minimum viable bigger somehow.

>>: For those who missed that, she said this is her company service, not something that we're going to be pushing. Um, we have a patient Bill who's been sitting on the phone. Operator, we are ready to hear from Bill. Hey Bill.

>>: At least it wasn't me this time. Great. Um, the, for me, this is all coming back to, you know, what do I need upfront versus, you know, what am I going to be able to get later on. I don't know, it seems like we keep talking about things that are fairly narrow in scope, talking about very specific use cases that don't necessarily tie back to each other. It's almost like we're looking at it from bottom-up versus top-down, not to overuse that phrase, but, you know, as long as we have the ability to have a concise and accurate list of components, I almost don't care how we phrase them, how we get there, I just need to be able to know that they're there.

>>: And can you spell out in a little more detail what a concise list of components means to you?

>>: Sure. So, you know, if we manufacture a product, we're software only, or in our case, medical

devices that have lots of software in them, um, you know, my goal is to ensure that for any particular device and/or software component, I have a list of sub-components, and not necessarily having to care where those exist within a product or where the hierarchy is, I just want to have the list. Where this becomes problematic is when you start, um, bringing in other third-party components, and whether it's complete systems or software packages, um, having the ability to force them to do the same thing, I suspect would get there over time, but, um, you know, we will lose quite a bit of data, or we won't get enough information to be, um, fairly accurate, if we don't push that, but how deep do we go?

>>: So, thoughts? No, I think that's, you've cut straight to something that, um, this room has been sort of going around for a little bit, which is, um, I'm using a third-party component, that component is using, right, you've got, um, transitive components, at what point do we say the person who is doing the compiling at point of time should have to go more than one hop, um, back upstream.

>>: It's like an experience, going after, you know, certain third-party providers of packages to get

what is in their software is worse than pulling teeth, honestly.

>>: Josh?

>>: So, to be a living, breathing daisy chain, um, of the 60 products or so we build, three of them end up in a lot of medical devices, including people participating and watching this whole thing. So, the FDA requirement makes them have to produce a CBoM, and if myself or my competitors cannot support them in that, in the format and hierarchy they do or don't need, they won't buy my stuff anymore. So, in a forcing function, the final goods assembler column in our graphic creates market requirements and procurement requirements on a compound part supplier, like a platform, like me, and what that's going to do is if I'm depending on open source products that are free, including single or compound products, that impair my ability to sell that medical device, then I'm going to have to replace them, right? So, there is a process of chaos right now, that the act of putting pressure anywhere in that process is now squeezing everyone else upstream and downstream in that process in what I think is a fairly healthy way. Not pain-free, but healthy way. So, that's why there's urgency to do this right away, because I need to know

which projects to get rid of, to replace, to get off of, to clean up, to update, so I can make sure all of my friends in this room can be compliant with FDA regulations.

>>: All right. So, I think this is, we made some progress, we've better understood the problem. I don't think we've fully settled things just yet. Does anyone have some last words on thinking about multiple hop lists, responsibility, and what this is going to look like, um, in sort of the medium term?

>>: So, we talked early in the day as part of the scope stuff on the concept of minimal viable product, which is, you know, the agile approach of do the minimum you have to begin with, um, but it doesn't mean you're done, there's further stages. We talked about moving on. It might be we start with the top line, but it might be we don't exclude to never get to second or third tier ever in the rest of history, we might just not make that be the first problem we solve. I think it's most important that we get back to the use cases and can whatever we create be made to work with the use cases we need, and if we do it that way, I think we'll get there.

>>: Is there a use case where the top line,

single depth is sufficient for you? Can anyone tell a story where going one layer deep is sufficient for use case?

>>: You mean one layer deep, one layer upstream in the supply chain?

>>: You are only including the top line of what you're directly including, and someone can give better terminology.

>>: Um, my understanding, my, um, I mentally came to the conclusion that we had some consensus on, in fact, that was okay, so let me try the whole thing real quickly. I am a software, whatever, creator of some kind, things I build in-house, I create a BoM for, everybody who I get a part from, I try to get a thing from them. Stop. Everyone else do that.

>>: The recursive process, where you're depending on your upstream component managers. I've got, um--

>>: I mean, in some ways, it's moot, right? Because of data, like, one thing that I've seen done, right, is that you have a customer who has a bunch of commercial products that has a bunch of open source components that it has to list in release notes because of licenses, and my company, I mean, they just put those

dependencies in monitoring, and you can do, with data ingestion, a full pack-down and get the transitive, but not hierarchical list, and monitor all of that stuff. The data's out there, so if you don't provide it in your SBoM, that may be fine, but someone's going to want it, and they're going to go out and get it.

>>: So, I want to make very concrete your assumption, because I think I disagree with you, but let's make it really concrete. So, if I'm a medical device maker, and I get thing works from Josh, and I list thing works version X dot Y dot Z and add some special sauce to it as the device-maker, and we just call it a day, then when there's an attack in the wild for JBoS blah, blah, blah, they look and say am I affected, and it says no, but that's because just below the thing works is the thing that we just said is being attacked.

>>: Well, so, typically --

>>: That's not deep enough.

>>: No, I agree with you. Um, what we experience in the commercial marketplace is that there are open source packages, because the open source licenses require that they be disclosed in the release notes, that those --

>>: Sometimes, yeah.

>>: Yeah, so, to the degree that they are, right, any component that you can get that's listed, you can get all the open source transitive dependencies for, like, that is a technological nonart.

>>: But I'm trying to see if you're saying the same thing or different than me. Which one are you saying?

>>: I agree with you, that the customer is going to want the whole thing. Um, I'm also saying that if you can't get the whole thing, because you can't force all your suppliers to give it to you, there's ways to derive it.

>>: Let me do, so, thing works uses JBoS, okay, so, um, you're BoM, you go one hop upstream, and when you tell someone, you got a thing works from me, it includes struts.

>>: So, I give --

>>: And struts gives you an SBoM that says I have Java, or I don't know --

>>: I'm pretending to sell to Jim right now. If Jim only lists me, then his customers won't know if they're affected by the next attack, even if I gave it to him as thing works. So, I do think if we only go

one level deep, we affect the am I affected and where am I affected case, the most basic of our use cases, the Jennings use case, right?

>>: I agree in the context of, like, commercial proprietary software, right? If you're just doing dev ops and you're using open source components, right, you can do all the investigation you want. If it's proprietary software that is not declaring its dependencies, and that gets folded into something that only declares its first-degree dependencies, then, Josh, you're right, you're kind of screwed.

>>: So, I want to make sure that we, we've got Duncan, Bruce.

>>: All right, so, I believe the way Ellen asked the question was is our definition of an SBoM, quote, only one layer deep sufficient, and I think the question probably needs to be modified, because, A, it depends on the use case, I think there's probably, hopefully, general agreement, that if the particular use case involves are you vulnerable to a particular CVE, you need to know everything, all the components at however many layers, so the question is can the one layer only model be used to also work in the all the way down to I have every component use case. If the answer is yes,

then we only need the one layer, as long as we show how that works, but I think there are definitely use cases that require you to know all the components, and, Josh, you know, gave a particular example that tends to be the classic example of, you know, whatever yesterday's attack was came out, am I affected, is a particular use case that you need to know every layer, the question is is however we're defining whatever we're talking about, I'm not even a hundred percent sure the question we're trying to answer, but can we meet that need with whatever we're saying, and if the answer is yes, then we're done, if the answer is no, then we need to do the multiple layer thing.

>>: I've got Bruce.

>>: So, it seems to me that the requirement is you provide both flat ones, and that should be the end of it. The requirement is you provide both flat formats, the direct one and the all the way down one, and then, you know, it'll satisfy everybody's requirement here.

>>: Is there any case where someone who would like the full graph, sorry, anyone who wants just the flat would not be satisfied by the full graph? Or is it a complete, is it completely subjunctive from a use

case perspective? So, in that case, like, wouldn't providing the full graph give the flat --

(Off mic.)

>>: Um, so, I think, if I've got this right, and, again, I just want to try to keep capturing where we are, there's a general appreciation that most use cases, we want to ultimately end up with the full graph, including the dependency relationships. What we are still unclear is is A trying to get just the B layer and hope that by including the B layer, the B comes with the C layer, or are we basically saying at time of build, A should really have a good idea of all the turns.

>>: So, I'm reminded of John Lambert's quote from awhile back, which is attackers think in graphs, defenders think in lists, and it seems to me that we are talking about wanting lists, or at least the SBoM that you're talking about earlier, um, attackers think, you said attackers think of the SBoM as a flat list, and actually, I think --

>>: No, I meant, um, the attacks don't care the hierarchy, they care if it's accessible.

>>: Right, actually, that's the key point I was going to bring forth. The attack surface is what we care about as that first layer list. I don't think I

actually do care so much about everything behind it yet, right? So, I think in the context of defenders thinking like graphs, we should have at least some purview of that first layer, so that we know how we prioritize what we patch or what we fix, because if you gave me a whole list, and there were tons of vulnerabilities and 40 of those components, but, really, only two of them were exposed, I would much rather spend my time on those two and not worry about the other 38.

>>: How do we get to that?

>>: So, I think the perspective is, I think, at some level, yes, in the future, we get some complete graph, but, honestly, I think I would really care about A and B. Presuming that B is what's actually exposed, right?

>>: They're completely independent, potentially, right?

>>: I think that's part of the problem that we're having, because the graph doesn't actually, ultimately, manifest what gets exposed, but I also don't want a long list of things I have to go fix, if I don't, if those things that are on that list aren't actually exposed.

>>: Yes, but different problem. So, yes, prioritizing, fixing the ones you're going to get popped with first and ignoring the rest is, yes, absolutely. I don't think anything we're talking about here is going to directly help you with that.

>>: No, I know, I understand that, but that's the tension that we're trying to figure out, because in the end, I actually still don't want a full list, because it just, it creates noise, right?

>>: Sorry, and just, again, I'm trying to be precise here, because you mentioned lists and graphs, so we have a couple of options here. One is the lists, or the flat, and that can be one layer deep, and a graph can also be one layer deep, but it doesn't make sense at that point, or your lists can be fully transitive, but without the transitive data, or it can be A calls B calls C all the way down. So, just trying to do this. Omar, you've been patient.

>>: Yeah, just one quick comment. Um, I think that we don't need a full graph, at least for my use case. As a vendor, I get components from other, not open source, but also other commercial vendors, where it's going to break, and if B is a commercial vendor, I cannot trace it back to when I get a repository or

anything else, I am dependent on that software bill of materials from that vendor, right? So, anywhere in this chain, if there's no transparency as a software, as an open source software, you will break it, you will be blind, right? So, that responsibility that I guess was mentioned before, that's the crucial part, right, because any of the, I guess these, I cannot see it from here, if it's a commercial component, then it's closed source, and it will break the chain.

>>: Unless your commercial partner believes in transparency.

>>: Correct.

>>: Um, Art, then Bruce. Sorry, Duncan, then Art, then Bruce.

>>: I was going to make a similar point, but back to the point on, um, I think we should not confuse the particular piece of information we need out of a particular pile of information and the presentation of the issue, like whether it's a list or a graph, okay? We're talking about the information content of for a particular block of software that you're procuring from A, um, what is in it is the information we're trying to come up with, and it might be for different use cases, the what is in it answer is a subset of what is

everything that's possible to be in it, you know, in his case of prioritizing, I only care about the ones with the high vulnerability, but to be able to pick out the ones with only the high vulnerability, you've got to have actually had the entire list to begin with, you've got to know which manufacturers used the Japanese seatbelts recently, even though, you know, it could be a Ford, it could be a GM, it has to do with who's using that particular supplier, which was to his point earlier. Um, so, you still need the collection of information of the entire set, and then the question is how do you filter it out, and I don't think we should worry too much on the presentation or the filters, we should focus on the information.

>>: And I think in this case, presentation is a type of data, so that is why I'm drawing this, right? So, this has no hierarchy data, this has the graph, the graph is shorthand for having the hierarchy data.

>>: The answer to that, which one of the three is better, might actually be use case-dependent.

>>: That's a great point. I've got Art, Bruce, Mark, Josh. Bruce?

>>: So, sometimes, if you're trying to figure out what you need to patch, the hierarchy matters.

The, um, issue that came out this year was not exploitable when it was in, you know, Eclipse, and Eclipse wouldn't put the patch out, because when they put the fix in, Eclipse stopped working. So, you know, in order to know that, I had to worry about the Baddach issue, I had to know what it was included in, and if it was included in Eclipse, I didn't have to worry. So, sometimes, we need this information for patching reasons, if that's one of the main reasons for getting this information.

>>: All right. Um, Mark?

>>: Um, yeah, I think the gentleman to my right captured my point, which is, um, one of the things I was getting feedback from my team that's listening in is we don't really have the customer here to tell us what they want. Now, I say that because I do know the FDA is here, but with all due respect, the FDA is not my customer, by any measure, so I think we need to be cognizant about the environment or situation where our customers are telling us what they want, and their demands that we incorporate directly into our software, as well as the libraries, so make sure that as we address very important issues, like medical devices and FDA, we don't get lost, that that is the dynamic and, you

know, I don't see the banks here, the U.S. government is already telling us what they want, and they're one of our largest customers, but they're already telling us what they want in the manifest. So, I think it's important to keep that in mind, that we are talking at different levels of application, and I think it does matter, to your point, um, in the graph that I can barely read across the room here.

>>: No, thank you, and by the way, um, for any of the customers that are on the phone right now, um, feel free to star 1 and chime in, because we do, in fact, have both public and private folks who would almost certainly identify as customers.

>>: Also, a quick comment, I'm your customer.

(Laughing.)

>>: And you're my customer, so, um--

>>: All right.

>>: You're very good at telling us what you want, by the way.

>>: Josh, and then Michelle.

>>: Yeah, I think he just hit on this in passing, but there's several customers in the room, just a lot of them are afraid to talk publically or identify themselves, but they're in the room. Um,

about 50 percent of the time, you go up to your chart, and I'm pretty sure I'm not the only one, I think I understand what you're saying, and then you go back up there, and then you undo what I just thought you said. I think we need crisp, maybe not in the moment right now, but very shortly after this meeting, we need a crisp, clear taxonomy of what we mean by your stratosphere there, because if what you're saying is the medical device maker A only has to list out their top tier vulnerability, or components at tier two ever, presentation or content, then I think we would miss most of the attack scenarios. If what you're saying is we have to articulate them with that relationship model, that's a different thing I could get behind, but one makes me panic, and one makes me not, and I still can't tell which one you're actually getting consensus on, so, it's not your fault, I just think we're all using the same words for different things.

>>: And I don't think there's clear consensus, so we're going to move on in a little bit. I actually think this was a constructive discussion, I don't think we've wasted time, it was really good to have these different perspectives, and we also learned, as you pointed out, that we need clear terminology, um, if only

we had a group that was positioned to do that, as well as a group that was focused on the use cases. It's like we have the right tools to solve this. Um, I've got Michelle, and then I've got anyone on the phone on this last question of thinking about how do we define the elements, um, so, go on.

>>: Thanks, Allan. Um, Michelle Jump. I do want to just kind of go back, and I don't even remember where it was, I was trying to jump in there. No, that's fine, but at some point, someone was talking about the actual use cases and whether some of these use cases might have more general top-level needs, and I just want to point out, as we're talking about the medical device situation, I wonder if some of these use cases that we're saying we need deeper levels, maybe because you're patching based on the information you're getting. In the medical device field, you should not be patching unless you have authorization that that patch has been validated from the manufacturer. So, a lot of time, at least, you know, and I'm looking at maybe some other folks who might be able to speak to that, um, more directly, but in my experience, um, that top level is a notification that these products may be affected, but it's not the end all be all. There's a

communication and a collaboration that happens between the manufacturer and the hospital, saying, you know, flag these products, because they might be impacted, but then the manufacturer's obligation is then to go identify those products and which of them are actually affected. That might take longer than we want it to take, but there's also the issue of, you know, patching without validation is actually against, um, what we're supposed to be doing in our industry. So, I think that's, and this is something that I have been very concerned about as we talk about getting this SBoM information out, um, is that what we don't want to do is start having people patch products that have not been validated to receive those patches and impact the connected hospital environment.

>>: I think our friends who are into micro-patching not withstanding, um, there is, yeah, um, I think most people would say that the alternative is not to roll out your own patch, it's to use other mitigation tactics rather than this, and there are, I think there's a pretty good suite of what I can do, if I don't have a patch on-hand, depending on the threat and what I've learned. Um, so, last comments on this question of what the elements are, and then since that

was so easy, um, I think it might be useful to have a little bit of a conversation about how do we share this information, what does the act of transparency look like, um, and I'm curious, if someone wants to, um, dive in on this front.

>>: We should do rapid fire speed mode. I think, just like you give a yula, you give a BoM. Every time you build, every version you ship comes with a BoM. That's a method. It's not the cloud method, but it's a method.

>>: So, can you be a little more precise?

>>: It is package would the deliverable software.

>>: I would --

>>: The cloud is different.

>>: I would completely agree. Every time you get a piece of binary, you should have an SBoM with it. I'd also say that every time you do a release upstream of a package, there should be an SBoM associated with it for people to do derivative work before they build it.

>>: All right. Is there, so, in terms of timing, I like that. For our device manufacturers, does that work? Sorry, mic, will you use the mic?

>>: It's either a direct BoM that's included with it or uniquely identified, that you can some place and get it so that it'll work for both cloud and non-cloud situations.

>>: One thing that I've had a couple conversations with, um, from the folks in manufacture usage descriptor, um, and, um, which is going through the ITF today, and one of the things about that is it says that a device, even a fairly lightweight device, can have a network-readable X509 certificate that a local router can read that will serve as just a pointer, and its primary use case is to serve as a pointer for a traffic profile, so this is great, it's, you have your toaster, and your toaster tells your router where to look, and there's a website that explains to your router that my toaster should not be talking to anyone, it should just be talking to the toaster. Um, but people have suggested that this could also be a great way to make sure that, you know, you can also contain a directory that says, by the way, here's my BoM, and that way, it just lives someplace not on the device, so we don't have to worry about, um, this, the weight of this. The other thing I'll point out is, um, some folks in the ITF have also been looking at SWID, they've noticed

that it's a big, chunky XML format, and they said if only there was a way to make it super lightweight, so that it can fit on an embedded device, and hopefully, Dave Waltermier will be able to chime in on that work. I've got, um, two fingers from Josh, and I've got Kate.

>>: All right, occasionally, people will say why don't we just do what MUD is doing. I was surprised you brought up MUD, but I'm glad he did. Um, and just point at a web resource, a lot of these medical technologies, vendors, and a lot of, even more IOT vendors go out of business, so if we're depending on a pervasively available web resource, that's maybe a bad idea, so to me, it would be an and, like, maybe you ship with something, and you can always point at the most current, vibrant thing. My guess is we're going to point at the resource for an ingredients list, I mean for a nutrition label, which is the enriched one, it includes mitigations and attestations, because we don't want to update the attestations every day, but the vulnerability list might update every day, so if we're hinting at the future, I think you also ship something with the product that you bundled, but you have a web look-up for maybe more vibrant, current, updated information.

>>: Thank you. Kate?

>>: Going back to the what else you should communicate is every time something's a patch comes out, you potentially want to be able to reference that.

>>: You want to not just give an updated version of the bill of materials, you want to explicitly be able to say this is a patch.

>>: This has been patched, yeah.

>>: And this has been patched, referencing --

>>: Referencing the original sources.

>>: Mic? Sorry.

>>: The patch has to indicate how it modified the original BoM.

>>: Sorry, we have, the question from Josh is one, if you have a new BoM.

>>: Yes, mine's referring to that. If we have a BoM that is the entire list of all the components of the software, are you saying that something in addition to that has to be, and this is different than the previous version by, you know, X, Y and Z? Is it sort of a release change thing you're asking for? I'm not exactly sure what you're saying, because aren't we, sort of by definition, saying we have the whole thing, so I, so, typical patches today, people worry about,

because you say, hey, I'm patching red hat, and what you're really patching is this particular, you know, heart bleed vulnerability or whatever, because it's buried three layers, but if we do actually have the entire piece of information that is made of A, B and C, are you saying you have to put in the deliverable the diff of this is different than the previous one? I'm not exactly following.

>>: I was just saying that there has to be a way it can be highlighted, okay? That there is a patch sitting there. You might deliver it at one point in time, and a patch may be sitting off to the side, and you may want to know whether or not that patch has been applied to your specific instance.

>>: So, can you explain to me the difference between patch and version and why, I mean, the purpose of the BoM is to say, hey, I switched component B from version one to version two. Is a patch something different than that, or it's just you're saying here's the ones I changed?

>>: Um, a patch might emerge between versions, it might be a fix for something that happens between versions. Go for it, Josh.

>>: I've struggled with this, so I'm not saying

I have the answer. I have struggled with this. People will patch a version without changing the version, which effectively changes the memory image, and that, so that is a historical pattern. I think implicitly in my vision of the future here, every change is a new version, instead of us keeping this ambiguous, gray area, unless you're a micro-patching fan.

>>: Well, are we drawing a distinction between a patch and an update by defining a patch specifically as something to address a security vulnerability?

>>: I think if the components change, you have a new version and a new BoM and a new unique identifier, and we have plenty of numbers that we could use. We don't do that today, but I think we benefit from more discipline and rigor and change control.

>>: I've got Mark --

>>: Let me just, again, I think this is getting into the granularity of particular use situations. Let me tell you, if we were to do that with our customers, there could be other chaos, just if we were going to change that kind of thing so regularly. So, I think you need to be careful about making assumptions about the nomenclature based on the situation of use in a particular environment, particular kinds of

customers. I mean, I've got colleagues on the phone who can tell me if I'm right. I think that would create utter chaos with our customers.

>>: Sorry, creating utter chaos by a new bill of materials --

>>: By renaming everything, every time you do a patch, I mean, because that's not the way our cycles work.

>>: Yeah, a lot of support language is based around major, minor patch, a lot of, um, charging for things is based around these, a lot of, um, internal policies of what you're allowed to do during a change window, you know, labeling it matters, so I agree with you about the historical nomenclature baggage, that's not to be ignored, I'm saying that the practice of a vendor putting stuff out that isn't uniquely identifiable has created other problems besides why we're here, and that, in theory, could have solved.

>>: But there's two things that are being put in the same bin. One is rolling a new component in, which, yeah, you should probably generate a new SBoM for that. The other is I fixed a bug in my code, and I put in a pull request, and that would be complete chaos, to your point, to rename a new version every time

you do that.

>>: Okay. I think, um, so, we're going to let the versioning question, give me a second, um, Art, did you have a very quick --

>>: Sorry, yeah. If you pull a, if you accept a pull request and change the code and re-package, you don't make it a P whatever release. Sorry, I guess, sure, yes?

>>: I want to check with our team. I think it really depends on the product. We don't just have one product, and each of them have very historical legacies and dynamics, so, um, I know that we're very careful, given who our customers are, who are themselves regulated and need some consistency in how they report it, that we're very sensitive about renaming or, I mean, we're very careful about when new versions come out, because it has implications for a lot of things.

>>: Sure. I didn't really mean red hat, but it's Josh's example exactly. If I change the software, and there's no stream, identifier that changes, then I don't know how the software changed.

>>: Yeah, and I think by any measure, red hat may be, um, we may not be special, but I think we are a unique vendor in that we, I think, do a very good job

of explaining to our customers what you're getting, what we've changed, because that's the nature of our customers, and they're under pressure, either internally or otherwise, to know what they're using.

>>: I'm going to get to Duncan in just a moment, unless it's, so, the model I use is the two fingers means you're throwing an interrupt, and I will prioritize you, but if you do it too often, you're going to get called on less and less, and, so, because I do want to get to folks on the phone. So, is this a, you want to throw an interrupt and chime in on this particular point?

>>: Yeah. I was just going to propose we add to the action item list of things we have to, you were making a list before of questions that need resolving. I think we're having a lot of semantic arguments, like on the word version verse the word patch verse, no, we just need uniquely identifiable software. I think we should solve that offline, but I do think we need to solve that.

>>: No, I really appreciate you flagging that, and we will document this, and there are working groups that are getting longer to-do lists. All right, on the phone, we've got Bill, then John.

>>: I should be off mute now, I hope.

>>: Yes.

>>: So, a couple things. With respect to, um, is a patch a new version or what have you, um, I guess I would posit that, um, you know, the actual software product, I'll use the word software here as product, would remain the same, but you would, I think Josh mentioned it, you'd have major and minor version indications as to what the patch level is. That is going to be deployment-specific, I don't think you can make that, um, part of the base product identifier, that would be messy. Um, that would be a major, um, version change, but when it comes down to identifying, you know, what patches exist within our product, that's always going to be site-specific, and we've got to take that into account. The other item kind of goes back to a prior topic with respect to what is in the bill of materials, and it was the concept of atomic versus molecular designations made, in other words, you know, the smallest component would be an atomic component, versus molecular would be made-up of, you know, multiple sub-components, just from an identification purpose, so you wouldn't have to worry that there wasn't any, um, other sub-components in play.

>>: That's really nice framing. I'm speaking just for myself, I like that framing of, you know, do you go down to the atom, or do you capture the molecule. So, thank you. Okay, on the phone, we have John Willis.

>>: Yes. Just wanted to suggest one, um, additional item for the, um, the structure of the BoM. Um, add another field called build number, use it as you wish, but, um, oftentimes, that's another number that's provided by the, um, whoever built the software, and sometimes, if it's re-issued software with the same version number, maybe that could be, um, purposed for, um, annotating that.

>>: Is there any comment from the room on build number being particularly constructive, or perhaps not? We have someone hovering in front of the microphone. Okay, great. No one is shaking their head wildly. Oh, Josh is.

>>: The more I think about Mark's point, it's not a technical point at all, it may be intended to be technical. I think there's a real hazard that needs real contemplation so we don't trigger an anti-body response, which is the business implications of changing the nomenclature, that live patch, custom patch thing has been a source of valuable ambiguity that

has kept, been the DMZ between what's in contract and not in contract, and we like having that valuable ambiguity, and whatever we do here should preserve the intent of some of that valuable ambiguity, less it be burned with fire by lawyers.

>>: Thank you, and this is definitely something we're going to follow-up on. Um, so, last comments on, and again, zooming up to the bigger picture of how this information is going to be shared, um, between the vendor, manufacturer, packager, assembler, and the person who's going to be using it, with a caveat that, of course, they're different use cases. So, if anyone has a use case of why certain types may not be good, that would be a good thing to throw on the table now as well.

>>: Just one question on, um, somebody suggested build number, and I was just kind of thinking, um, that a date might be a valuable, um, piece of information there, and just thinking of use cases, I may want in a BoM to know, you know, because you can't have a new, let's say BoM for every single time that you tweak a piece of software potentially, right, but what I may, as a, um, user of software, I may want to stipulate that, um, you know, I don't want the BoM to

expire beyond, maybe, a certain amount of time, you know, so I want it to be relevant, or, you know, current within, let's say one week or one month.

>>: So TTL on the BoM?

>>: What?

>>: Time to live. You want an expiration date.

>>: Yeah, perhaps. Yep. Or a date of when, um, the, um--

>>: Are we conflating a BoM, which is just the straight-up ingredients, never changes, with the more femoral or transitive state of CVEs and things in the nutrition labels? Are we combining those two in your question?

>>: Um, so, yeah, no, I was just thinking, um, that, um, a BoM is taken at a specific point in time, right, and the components that you've got in your software might have changed at various --

>>: Then I'd say it's a new version with a new BoM.

>>: With every --

>>: Every time. If your cloud service deploys hourly, you have a new BoM spit out hourly.

>>: If that's, yeah, I mean, if that's the solution, it sounds fairly heavy, but --

>>: We're going to go to the cloud side in a second, but certainly, for local software, um, I think the notion is that, as I've heard expressed by a number of people, as built, and, so, every time you build, um, but I just want to follow-up on the first suggestion you had, which is having a time-stamp. Is there a value in having a time-stamp? Duncan?

>>: I think we could get very hung up and spend a really long time on arguing what the word version means and time-stamps and patches. I think Josh said it, I'm not sure I'm going to copy it exactly, I'm not going to remember it exactly, I think we need a uniquely identifiable identifier of the software you're talking about, that if the software changes, you're now changing something or another. Now, certain use cases might use semantic versioning and major/minor and whatever, and certain use cases might use the, you know, red hat's been doing what it's doing for a really long time, so I'm not proposing they have to change their versioning, okay? Um, whatever meets the criteria of I can tell the software changed is all we need to deal with, I would argue.

>>: I think we've got that in a lot of ways right now with just the hash, a hash on the evidence you're

putting out.

>>: And again, I'm just saying just sort of leave it at that and get on to other things to argue about.

>>: Thank you. All right, um, we have an hour left today. I think, again, this has been useful, to sort of tackle some of these issues and show that even something that, in a big picture might be seen as simple, once you bring the experts into the room, they show that there's no such thing as an easy problem, because all the easy problems have been solved, and, so, I'm very glad that all of you are here. So, let's tackle something that we've alluded to a couple times, and we're going to probably, again, return to our use case discussion, but let's talk about, when we move away from the on prim deployed software, um, and we're going to use the cloud as the shorthand. Obviously, it's a very complex set of, um, architectures that we're trying to capture. I'll put two positions at either end of the spectrum, and I want you to sort of figure out where you like this. One is from, for most users, a lot of the use cases that we have for on prim software bill of materials, we also derive in a pretty straight-forward fashion from a bill of materials about

a cloud-based product.

So, that's the full-on, yes, absolutely for SBoM and cloud, and we don't need to make any changes to how we're thinking about it. At the other end is an argument that between ownership responsibility and between the fact that modern cloud deployments have just this incredibly diverse and complex architecture, how we think about a bill of materials is wildly different and may not, in fact, have an inherent value for some of the use cases. So, that's the absolutely not, and of course, and I'm putting these out there as, obviously, toy arguments. I'd like to hear how you react to that. Yes?

>>: And I hate to overcomplicate things, but there's something between as well that is a hybrid approach, that is a cloud service is actually updating hardware on prim device, and now it's providing some type of bill of materials that it recites there, and it changes also all the time with some other services, APIs, you know, whatever the case might be.

>>: All right, so, yeah --

>>: And actually, we're connected to one right now.

>>: This gets to the notion that the line

between cloud and on prim is, in fact, quite blurry, and if this is, this might be a different use case rather than a middle position of the spectrum. Um, yes? Duncan?

>>: Could you or somebody say more on the cloud use cases you're thinking of? Because in my mind, there's a difference between I'm using a service, and it's sort of the service provider's, how we built this software and whatever, I don't care if he's got the vulnerability, it's his service, he's got to worry about it, verse the I'm putting something in the cloud, and I'm in charge of the components and doing whatever, and then it's more like the prim case to me. So, the host type of thing, so what exactly are the particular use cases you're trying to solve?

>>: Got another two fingers.

>>: A comment, not a solution, but Nest created a nomenclature of all the community clouds, hybrid clouds, service, you know, softwares for service and everything. I think we probably have to look at them to address them, this is out of scope, this is in scope and everything, but both from an infrastructure and a software as a service, think about, I'm only pointing fingers to myself, you know, WebEx, whenever you're in

the WebEx, that's a service, wherever that recites, that also should have a bill of material to support you, right?

>>: Yeah?

>>: This is only one of our use cases, but it's the primary use case. Um, to answer am I affected by a new attack, every time the software and memory image changes, regardless of where it is or who owns it, it changes my attack profile, or potentially changes my attack profile. So, I think, maybe it's just such a basic point, I skipped it, but to me, a change in the attack surface or attack, you know, stack is a change in an SBoM, and if they're sutured together in our heads, then it's not about what is it, it's more about how do we deliver it, and that's going to be, like I said, is it an API, is it an interrogatable thing, is it a dynamic thing. When you look at some of the really forward development environments, they're drag and drop, as you imagine it, you create mash-ups of, you know, micro-services. There's not even an architecture to it, so no one's going to, like, deliberately retroactively go back and make an SBoM, it's going to have to be, you know how they say security is not composable? SBoMs are, right? If we have every

single piece at any part of the stack, whether it's Amazon hosting your pass middleware or your custom secret sauce that's at the sauce layer, the composable nature of that attack surface to an attacker, we should be sutured to. Now, I realize I mixed attack surface, which is column C, with the software composition, which is column A, but I think we have to think that way. It's less about what's in it and more about how do we access it, and how often do we access it.

>>: So, I think the way that we would look at SASS services would be, or cloud, but SASS in particular is that the threat model that we produce for that SASS service assumes that they may get compromised through whatever software that they have within their stack, and it would be incumbent upon us as the users of that service to ensure that our content is well-protected and dependent of whether or not that entity is compromised. So, the premise here, again, is if, um, regardless of what their SBoM is, um, we're already designing our security controls, um, based on the threat model that they are compromised, whether it's an administrator or some external adversary. Okay, the other perspective, um, is in terms of, the other type of inventory that we do is we try to inventory all

the APIs that we use externally, so it's a different type of SBoM, if you will, but it's one where we're trying to understand what is our exposure as it relates to all these different APIs that are being made available to us, and we're consuming that for our own purposes. Um, not an easy task, but it is one that, um, is part of our own internal SBoM of if you're billing a piece of software for some purpose internally on premise, what, um, what components are you pulling in in the form of APIs.

>>: Thanks. Um, and I'm going to ask, do you have, therefore, some opinion about whether or not this group should try to tackle this, or is this something, and you can say no, I don't have an opinion, or do you think this is sort of --

>>: I have an opinion. It's, I would agree, it's the latter. It's a different enough use case that I would not try to make it much more complicated. This is a hard enough problem as it is. Let's tackle this one first.

>>: Good. Thank you. Kate? Nope? All right, we have Bill on the phone.

>>: Um, seeing as the web cast is, um, delayed, I can't see the two fingers, so that's good. The aspect

of web services, um, micro-services in particular, um, I would probably argue that those are going to be treated as atomic items, only because, you know, the provider is going to be responsible for maintaining those. I wouldn't, as much as I'd like to know what's in them, I almost don't want to have to care, because I'll just treat those as something else that's within my application, especially for those, um, single-page app, where it's micro-services on the back-end, your BoM is going to be fairly simple, but, um, when it comes to, you know, vulnerability management and assessment, that's the level at which I think we're going to see things. We're not going to see things underneath the covers for them, we're going to see it listed as an issue with, you know, lambda or S3 or something like that versus the pieces that make them up.

>>: Okay. So, we're going to see, there's some less visibility there. Um, further thoughts on this particular issue? Again, this doesn't have to be the last time that we tackle this aspect of it, but it also can be, if enough people stand up and say let's stop, I'll leave this outside of scope. This can be the sort of thing that our use case group can, um, sort of start to tackle.

>>: Not knowing NTIA as well as you do, can we have items where we agree not to wade in too deep, but we have, like, an opinion on? Like, we think it might look like.

>>: It's your party, you get to have, um, you know, if you want to write-up a draft of here's why this should be thought about by future people, that's great.

>>: I mean, more specifically and less sarcastically, you know, the split decisions in, you know, court cases, right, there's a dissenting opinion, right, could we articulate what's the best case for treating this no differently, what's the best argument for treating it differently, let someone else, at a future time, go pick up our thoughts, right?

>>: No, that is absolutely something that you can do, which is to say here are some issues where we did not have complete consensus, but we think they're important enough that they should be considered regardless of what the outcome is. I think documenting, again, the expertise of folks who have thought about this and thought a lot about the benefits of bill of materials, as well as, you know, what, you know, what makes this aspect of business, and the more you have behind that, the more likely it is that someone

is going to pick up, um, pick this up and work with it.

>>: Let me clarify then. It's an important issue. I think the hard part of this is less for a cloud provider, the hard part of it is less the software they use and the fact that the cloud provider is not you, right? So, I think in the context of what your concerns are, in the hierarchy of concerns, you should be more concerned about the provider itself turning malicious than the software going bad, right? And, so, um, while I agree, that it is a concern, um, it was more of a hierarchy of ratings, if you will, and that's really manifested through the threat model that we perform. So, the threat model shows that a malicious insider in that company is going to be far more likely to be, um, damaging to me than the fact that they may use struts.

>>: I'm going to ask one more question around this, which is if you are wearing your customer/consumer hat, do you want, does it mean anything if your provider says I have a bill of materials, or, no, I cannot produce that? Regardless of how you would use the data, just as we've talked about, you know, channeling my inner Steve Lipner, if I can invoke him, a lot of the value, just does the vendor have the capacity to generate the space. So,

as you think about this, is it important that your service provider, whatever that looks like, again, acknowledging this is a very diverse space, be able to produce that list, be able to produce that internally and manage that internally? Mark, then John.

>>: Um, so, I think, inevitably, the cloud is going to be an environment that we think about here, and I think it will depend um, and I don't know if Kent is still on the call, but my one take-away from the first meeting was that we were going to start with the simple use cases first and then think about the cloud. I would continue to think about that as a work stream. I will say that the, in this context, one of the differences I discern between the traditional on premise or even private cloud approach versus a public cloud approach is that this discussion of transparency is largely a one-way street when you talk about the traditional. In cloud, it is a two-way street, and I think that until we have a discussion about what is the obligation of the cloud provider, public cloud providers, in providing API, you know, documentation, everything else, the discussion of an SBoM for, um, software used in this environment is not going to get very far, and I think it's a very different dynamic in the public

cloud versus the traditional environment. So, I throw that out to make sure that we, you know, make progress in things that are manageable and controllable at this point, recognizing we'll get down the road on the others.

>>: Josh?

>>: Your specific question of as a consumer of a cloud service, I think, yes, it's meaningful to me. I'm a power user, but it's meaningful to me in a similar way according to the vulnerability disclosure program is meaningful to me. It tells me they're more likely to hear laws from good guys before bad guys. It also, um, indicates a level of maturity that may stand them apart from their competitors, but number three, and a very measurable way, if they have an SBoM, my hunch is their mean time to remediate will be marketably different than if they're blind every time.

>>: Quick interrupt, and then over to Mike. Yes?

>>: I would never interrupt the note-taker. My point of APIs was very specific. I think most public cloud providers will be glad to provide you with the variety of APIs they have. The question is are they providing you with complete, accurate, and full

documentation of their APIs, so that the utility of an SBoM can be meaningful. If they don't do that, then SBoM, I think, has marginal utility. I'm being extreme here, but that's the difference between the two, traditional environment versus the cloud environment.

>>: It's really helpful. Thank you. Mike?

>>: My point was just covered nicely. Thank you.

>>: Oh.

>>: I just realized I'm a cloud provider. I provide banking services, and I don't know if every customer would want my SBoM.

(Laughing.)

>>: Right?

>>: But if I had a choice in two banks, one that's capable of internally measuring their own supply chain and hygiene, versus one that can't, I'd pick, I think --

>>: You'd pick one that would have, right.

>>: I don't see it, but I would need to know that you're on it, but I also am FDIC-insured, or you are rather.

>>: Anyway, I'm just putting it in context for myself, where if I see myself as a cloud, a SASS provider

of banking services, um, I think I would actually not, yes, I would want to certainly, just internally, good hygiene, you know, all that kind of stuff, I want to have good SBoMs and stuff, but I don't think, if I were to look at sales force and say, hey, sales force, give me your SBoM, I would say that seems just completely out of bounds, just as much as it would be for me to offer it to a customer.

>>: Yeah, so, there's a, um, I think a way out of this, so to speak, is, um, on prim, product and software, there's a vulnerability in something, I have to go track it down, decide on my risk, go patch it or find it or something like that. SBoMs very useful. Knowing that my cloud provider has something buried three layers down doesn't really do much for me, they have to fix it anyway, I'm at their mercy, knowing, if I know they have a bug and they're fixing it, it doesn't really matter to me where it is in their SBoM. I want them to know, they should have an SBoM for their own purposes, but I get marginal utility at best out of knowing what they're doing, especially if they're changing it hourly or four times a day or something.

>>: Is the demarcation point operational in responsibility? Like, it's meaningful to someone with

operational responsibility?

>>: I mean, this is full management case, right? I have to take action, I need the BoM data to figure out what to do better. If someone else is, if the cloud provider, if the bank's liable for messing up and it's their problem, they have to have their own, but I don't, their down stream doesn't care.

>>: I have invoked Steve Lipner, and like that, he has appeared to us on the phone in the ceiling, so we're going to hear from Steve, then JC, then Duncan.

>>: Yeah. Hi Allan. You raised a, you brought my name up, and that, I figured I better, um, better call in and say something, so you'd know I was at least still here. I've been having, the web side of the conference is so out of sync with the audio that I'm having a tough time, you know, sort of knowing who I'm hearing or what's going on, so this is somewhat anonymous, but I think you summarized my point, um, accurately. There has to be, you know, in my view, the SBoM is useful to somebody, not to hold up an SBoM and say, gee, I have an SBoM, but to use it to actually do something within their scope of activity and effective. Um, so, you know, do I, as an end user customer, do I care about, um, about the SBoM of the software that I'm

relying on? Maybe not. Do I want my supplier, who created that, um, that, um, that software, to have an SBoM that they, um, rely on and can use to manage the software? Yes, preferably all the way down to the sand, and, so, I think that's sort of a rule for thinking about SBoM, you know, who can use it, and what can they, can and will they realistically do with it. Um, it's not just something to, asking somebody to show it to you, to demonstrate that they have it is maybe not the most efficient way to, you know, to deal with it, but knowing that the organization that's maintaining the software, updating it, releasing the updates, evaluating vulnerabilities, deciding whether some action needs to be taken, knowing that they have an SBoM and are actually using it to make those decisions is very important. I hope that makes sense. Thanks.

>>: Thank you. That's, um, always great. I've got, um, JC, and then Duncan, and then we'll move on.

>>: So, um, to Art's point, it is an incredibly common situation, that vendors have vulnerabilities that they do not address or will not address. Um, and, so, I think the case for transparency here is you can't make the vendor do anything, but is there any sort of

remediation you can do, um, on an infrastructure level, or, you know, put something behind a load-balancer, like, is there anything you can do in your systems security plan to remediate this vulnerability, which the vendor or their supplier is not going to fix? And, so, you know, that's the operational question.

>>: That's really helpful. Duncan?

>>: Thank you. Um, I think we might look at our terminology a little bit, and I think prim and cloud might not be the best choices. Um, a lot of what we're talking about when we say prim have to do with a device of some sort, have to do with your buying something, a device might be a virtual device, it might be a collection of software, and there are cases in the cloud where you're, again, buying a virtual device, and in those cases, I think the SBoM, the exact same arguments we made about everything else applied just as much, and there are other types, other times when you're consuming a service, and there, like was just mentioned, sometimes, you can't do anything about it, but there's still value in knowing, because also, you're doing risk management, or you're doing risk management for the vulnerability use case, and in that case, that's when you'd want to do that, and again,

maybe you don't have choices, maybe you go to a different supplier, maybe you can put in controls or something. Um, if it's the cost case that we mentioned, or in use cases, sometimes you also do SBoMs, because they save money and they're good things, those are less applicable to that particular case. They are applicable to the device cases, they're less applicable to the interface cases. So, I think it does come on, um, on terminology, but I just think if we avoided the prim and cloud, we might sometimes make it easier, if we got into more specifics of which use case we're talking about. Thank you.

>>: Excellent, and I love this common theme that's really been at the core of everything today, which is, comes down to the use case and work up from there.

>>: Yeah, I'm not sure this changes anything, but did you say you might be buying a virtual device? And if you're talking about a virtual machine, isn't that a service?

>>: No, a virtual device is different than a virtual machine. I mean, a virtual machine is you're buying someone's compute power, I mean, it's a particular type of virtual device.

>>: That's an example.

>>: But I, I was chief architect at AT&T, I used to spend millions of dollars buying from Cisco and other places virtual devices, which were basically their software running in my cloud. In fact, I'll say most of the network cloud you're talking about today is, today, mostly run by buying software from people and running it on hardware that you bought from different people, and that case is more like a classic device case with an SBoM, and you want to know everything and everything else.

>>: Okay, that answers my question, because my question was aren't you buying a service.

>>: Well, when you are buying the service, that's when this gets iffy, and it's harder, and I can't wrap my head around it quite as much, but particularly in the device cases, when I'm buying a compute infrastructure and putting software on it to do a particular function, then that software bill of materials matters a lot to that chunk of software. When it's an API to something, it's a little bit grayer in my mind, what the SBoM buys you about the API use, other than the whole risk management of I at least trust them better, because they know what their software is,

and I can look up, if they give it to me, yeah, I know they're susceptible to this bug, and they haven't fixed it, so there is still a, there's still a value there, but it's a different value than the more direct control device or virtual device case, where it's, you know, my pacemaker or my, you know, virtual router in the cellular network or whatever.

>>: All right. Um, I've, I do want to move on, but we've got, um, Steve back on the phone, and then Josh quickly.

>>: Yeah, so, a woman spoke right after me and said, you know, talked about relying on vendors that don't, you know, that, um, when there are vulnerabilities reported, they don't remediate them, and my preferred or recommended solution to that, I'm not sure that's an organizational position, but my personal recommendation there is get another vendor. I mean, that's sort of the long-term solution to that problem. You know, my perspective on this is based on the expectation that vendors faced with, um, with vulnerability reports will investigate them and remediate them, and if you, if you've got a vendor who won't do that, I think you're, I think you've got a bigger problem.

>>: I mean, I feel it is not too far of a stretch to say that Canes is still right in this case, that in the long run, we're all dead. Hopefully, not in the ICS space, but it's a little too on the nose. Um, Josh, then we'll move on.

>>: So, what I like about this chat is there's a lot of critical thinking going on, people that started at one point are really wrestling with this. I think one of the things that's been on the tip of my brain, and maybe you're getting at this as well, if you get past the technical layer again, if we use, like, a car as a metaphor, the final goods assembled, there's some liability associated with harm that's clean. When you go to software, there's no liability, and lots of attempts to make sure we never have it, and I'm not trying to go to the L word. Cloud and services are somewhere in between, because there's much more rigid contracts and service-level agreements and spheres of control, in part due to cloud security alliance and a lot of the A6 work, so I almost wonder if one of the reasons we're struggling with this is our technical part, it's a mixture of technical and accountability, responsibility, and I haven't solved it, I'm not going to solve it in the course of me un-muting my mic here,

but that's part of the, our confusion, right? Yes?

>>: Definitely.

>>: All right. We have, um, about a half an hour left. I do want to get into some of the logistics, because I think this is something that, um, this assembled brain trust can tackle and see if there's a better way we can self-organize, but before we do, I want to get back to something that, again, has come up a little bit here and there, which is independent of the bill of materials, and, you know, we're leaving this part aside, um, there may be some real value in hooking other types of communication that explain sort of this vulnerability idea, and, you know, I think from both perspectives, from the vendor and the customer, if it is, if there is something that is going to trip a naive sensor, and it's not going to be something that the customer needs to care about, no one wants to deal with this, so is there a way that we can sort of find a way of communicating issues around, and I'll use the term, and it's not a great term, but I'll use the term a false-positive, um, against the naive strip? We had an intake of error. So, the question is if we have a software bill of materials layer, and both of the tools, um, and the formats that have been proposed, um, both

SWID and SPDX, allow pointers to other things, do we want to have a conversation that we can start now, we don't have to finish it, but we can start now about what those pointers may be? Um, do we want to say let's start thinking about how we would hook a discussion about vulnerabilities or mitigations, or to use Josh's word, attestations, to this data plane?

>>: All right, are you asking if people will, whether we want them to or not, cross-reference national vulnerability database for potentially exploitable vulnerabilities?

>>: I think what I'm saying is if we assume that there is going to be at least some use of this data --

>>: It's already happening.

>>: To flag against this type of information, do we want to allow, or do we want to enable further communication from the vendor to the customer, hooked to the bill of materials, that explains or gives otherwise mitigating information to say this is why you should feel comfortable ignoring this blinking red light? I'm looking at, I want to start, Bruce, I feel terrible, I wanted to make sure that we get your engagement on this, and I know a few other people have had some thoughts on this.

>>: So, when I put up those three columns, the one I'm most scared about is the middle one, right? It's kind of like the middle of the road, where all the cars hit you, right? We're on the blind side of the road right now, where we don't know enough. If we stay in the middle, we could have a lot of these difficult and potentially noisy conversations about potentially exploitable vulnerabilities, most of which won't be exploitable. If we push through, I think this is future-proofing the standard of format, which is creating this space, the name space, or the fields, for me to make a claim that says I have tested this, I have hardened this, I have put mitigations in place, my configuration, this is not vulnerable in my implementation, something along those lines. So, I think if we allow our sales to stay in the middle ground, we'll have lots of problems. If we push through straight to the third one, we'll be better off. I don't think it's a lot of fields, but it future-proofs the format.

>>: Can I push on you? Can you sketch out, what would that look like?

>>: So, um, I'm going to butcher, Mark, hopefully, he's on the phone, but his source clear

stuff, he showed things, said out of a hundred potentially exploitable vulnerabilities, a single digit percent are actually in the control flow, and I had to push back and say there's other ways to exploit, but let's just say ten of those hundred are actually exploitable, so if someone says go get, go fix all the other ones, that could be a lot of work.

>>: And I think we all, what would that --

>>: So, I would put in a column as the vendor, or the supplier, um, we have threat model tested, etc., this is not exploitable in our code at this time, blah, blah.

>>: I like it. I've got Mike and then Art and then Bruce.

>>: So, I'm sort of picturing myself as a product vendor, asserting that this particular exploit cannot be used. Okay, I just wanted to highlight that element.

>>: Bruce?

>>: So, certainly, in the general case, it's difficult to do that, but let's take a real example, and an example I'll pick is log for J vulnerability, it came out this year, and at Oracle, there was 450 products that used it, I don't actually think any of

them were vulnerable, but certainly, it's less than 5 percent, and that's because they don't use the feature that was vulnerable in log for J, and we know that, because there's no way to configure it in those products. So, you know, in order to make this work, you know, there's two things, I think. One of them is the naming issue. So, I know that when it says open SSL or whatever, I know what that is, or when it says Oracle product X, people know what that is, and the other one is this issue here, where the vendors need a way to say this vulnerability is not exploitable in our product with this component, so that if Eclipse could have said this in a machine-readable way that said we're not vulnerable to Baddach, you know, vulnerable CVX, everybody down the lane from that would know that, and the case of this one, um, Baddach inside of Eclipse, Eclipse is inside of Java, Java is inside of the Oracle database, the Oracle database is inside of some Oracle products, everybody all the way down there has this problem of explaining to their customers all the way down, and lots of them, that they're not vulnerable, and we've done that. If I could just put this online, a whole bunch of people, and when we're putting out patches for log for J, we've got about a hundred products that we're putting out the fixes with no other fix but that one, so people are applying fixes that is not improving their security at all, and anytime you do any change, it degrades things with some probability, so it's doing no one a benefit, you know, and all that will do is cause people to not put on better, you know, important fixes, because they can't handle so many fixes.

>>: Thank you. Um, I've got Kate, then Omar.

>>: Yeah, I'll just, um, I agree with the problem of the signal to noise issue, when you've got a lot of the false-positives. One of the things I'm wondering is to what extent should we looking at tracking and recording config information from the bills? Would that be helping to explain that something is not potentially exploitable?

>>: Yeah, so, this has to be separate, right? Because you put out a product, and then if it had Eclipse in it, just go back to the same piece I've used the whole time, somebody later discovers there's a vulnerability, so this has to be independent of the BoM. I'm not claiming it should be part of the BoM, in fact, it can't be part of the BoM.

>>: There are myriad qualifiers willing to

give. One could be this is off by default, you're at risk, if you turn it on. I mean, we could say anything we would want to say, but communicating our mitigation. Also, it has the benefit of, if we're lazy and we just make something up, we put in writing that we did something, and so we could be held to our claim.

>>: Omar, then Duncan.

>>: So, I'm definitely in between both. So, we score, not score, we disclose worst case scenario, so even if it's not truly potentially exploitable today, if somebody configures it, you know, differently and everything else, I'm there with it, the challenge that I see, putting that into the BoM, and I'm not against it in one way or the other, um, yeah, at a nutrition level is the time, again, the chicken and egg thing. So, if we're building this or pushing this into the build time, then I have to actually put all the different CVEs and everything else that is not applicable, we actually tried that in one product, it failed miserably. We used to put that in a release note with all the CVEs that potentially are in that, and the customer said, you know --

>>: I think these attestations probably aren't at, what's the phrase Kent uses, as built?

>>: As built.

>>: They're probably living resources on the web somewhere.

>>: Yeah, I think the notion is there's a column, which is to say it is condemned of your software bill of materials, versus I can imagine there is a pointer, um, that could be, you know, if not actually standardized, then certainly, um, you know, de facto deployed in such a way that everyone knows that a URL structured this way, um, indicates that there is something good waiting on the other side. Um, Duncan, then JC.

>>: So, again, getting in terminology, so we, I think, are stabilizing on this thing called an SBoM, and we're now discussing, I thought, whether to add another column to it associated with things, and you just, but that's a different thing, a nutrition label is a different thing than the ingredient list, so the SBoM is the ingredient list, I don't think we should add another column for the CVEs. I do think the CVEs need to reference, or a different thing we come up with needs to reference the relationship between CVEs and SBoMs. This particular SBoM was built on this particular code on this particular day with this

particular set of software, and it is vulnerable to this, and it's not vulnerable that we know of to this, this, and this. That relationship is a different thing than an SBoM. The SBoM is the thing released to, I'm not against nutrition labels, I just think it's a different thing than the ingredient list, and the SBoM is the ingredient list.

>>: I think that there's a case for yes, and in terms of the live evidence, because our, again, operational experience is that you need that statically generated SBoM for audit. So, it's not about whether you're green now, it's whether you can prove that three months ago, you were compliant, and, so, you do need to be able to archive this stuff in an immutable way, just a different use case. Um, to your point, you know, the SBoM is different than the, essentially the assurance package. Um, in practice, those things get combined, but they don't have to be.

>>: Thoughts? Responses?

(Laughing.)

>>: That's it.

(Laughing.)

>>: All right, so, what I'm hearing, unless, is there --

>>: Well, just for the note-taker, this assurance package, at least in my graphic, was the nutrition label.

>>: So, I want to avoid using the term nutrition label, because that is being used a lot by a bunch of other people in this space to mean wildly different things, so I know that there's a lot of value in that metaphor, but if we're focusing on a very concrete issue --

>>: How about the framing, can I nominate the framing guys take an action item, which is whatever words we change to at any given time, we at least know what word refers to what content.

>>: Yes, which loops us back to where we are, which we've got some questions that we've walked through this afternoon, yes?

(Off mic.)

(Laughing.)

>>: So, we've walked through, um, a number of things this afternoon, and it looks like a lot of this stuff is, in fact, going to end up on the framing and understanding the problem plate, in part because it is around understanding the problem. Um, the, we've still talked about the general idea of what are the

components, and that came up with, we sort of got a couple of dimensions here of both does it come with the semantic data, or is it just a list, graph verse list, and then who is responsible for producing, to what depth I think that's the first thing, and I don't think there was a perfectly clear consensus, but I thought we did a good job of exploring the space, and we'll make sure that there's some notes that are available for the group. The second thing was on how we're going to convey the data, and it was a productive discussion, and we also learned that we do need to have, probably, some clarity around versioning was the thing that sort of popped up. We also have acknowledged that naming is a known hard problem. I don't know if a particular solution was advanced, but this might be something that we want to continue to take up and poke at, and one of the things that we might want to do, and we can work with you on this, is collect resources, because there are a bunch of tools out there, people have tried to tackle them initially. All of these are being put on Michelle and Art's place, unless someone wants to jump in. Naming actually might be a good fit for the standards and formats group. We've got some hesitant head-shakes. Um, before we, and then, of course, we talked about cloud, and I think that boiled down to really thinking through use case, and, so, that might be something, if it fits into how the use case group is approaching their problem, does that make sense, to sort of think through and get a little more clarity in how a bill of materials might manifest in different types of use cases? I've got a hesitant look from Josh. So, sorry, first, use your mic.

>>: I don't think that conversation can happen without also looking at the differences of contractual accountability/responsibility that do vary wildly between cloud and, yeah.

>>: Agree completely. I think that's something that a lot of people mentioned. Some people advance there are going to be uses, um, where having this data could be useful, and others said, no, it's not under my control, I think if anything, there was some discussion to say, perhaps, we should package up what we think about it and leave it as an unresolved. In fact, that was the idea of a very smart, rather good-looking person, if memory serves, who sort of proposed that. Um, but I think it's still, it's one of those things that comes down to use cases, and then, lastly, what did we just talk about? Oh, the sort of

vulnerability information, and is that going to be, um, something that, um, our understanding the problem folks want to start to begin to look at? Probably.

(Laughing.)

>>: Yeah, I'm not completely willing to own it, but we'll add it to our big question list.

>>: So, this gets us to the core of how we're going to move forward over the next few months. Now, unfortunately, many of you have families and jobs, and both of those take up a lot of your time, especially with the holidays, and, so, we want to acknowledge that, you know, things are going to be busy and hard to schedule over the next few months. Um, from our part, we are thinking of having an in-person meeting, the default of being in Washington, DC again, somewhere around February 20th to 25th or 6th or 7th, somewhere in that window. That is either a week and a half, or the week before RSA, and we're trying to do that, so one thing that's very useful is from my mapping, there isn't a major information security conference, and I haven't heard about a giant, um, medical IT event in that window, but if there are, we want to make sure this idea, this date is shared as quickly as possible, so that you can block your calendars, so I want to get some thoughts

on having that window, and then our next question is going to be does it make sense to have a virtual meeting in January as sort of a short-term progress meeting. So, Josh?

>>: I kind of hate this is coming out of my mouth, but I hate the idea of coming here before, um, is commerce dead against doing a stakeholder group where we're all, many of us will be in San Francisco during that week? Like, um, we did this in Chicago for CVD, we did it, you know, can we do one on the road?

>>: So, we are happy to go on the road. If someone wants to host a discussion around a major event, that's great. The problem with hosting something around a 40,000-person conference is finding a day during the week of RSA where folks might be available. That's where I'm a little worried about trying to find that time. Um, certainly, we can have smaller discussions during that time.

>>: Just a point of clarification, RSA doesn't schedule speeches Monday for purposes just like this, and they tend to put crappy ones on Friday for similar reasons. Now, I'm in, like, six workshops on Monday myself, but the point being this is the kind of place where people can have existing travel.

>>: Put you on the mic.

>>: That week, there's also going to conflict with embedded world in Europe.

>>: Okay. Thank you. I think that window actually spans a weekend, so I think we tried to capture, um, sort of either the, um, either the 20th is a Thursday, maybe, 20th is Wednesday, and so this either gives us two weeks before RSA or the week before RSA, and, so, but the whole thing is embedded. It's almost like it's really well-situated. Okay, um, so, that is one of the big conflicts. Um, again, if someone wants to say, you know what, we should have this someplace warmer, um, it's easier to do this, again, co-located. I'm not sure we're quite ready to co-locate it, um, at a European conference, that might be a stretch. You're kind of looking at almost all of NTIA's budget right now. Mic. Sorry.

>>: I think you could pick a date and, you know, something about size, then there's a lot of people, Cisco and us, or, you know, maybe even red hat or somebody can have a good chance of being able to come up with something. Um, they have, you know, stuff that's big enough, we think we do, I don't know if we can get permission or not, but I certainly am willing

to ask for it, if that's what people like, but I don't want to have to pick the date.

>>: Okay. So, we're going to go with, and let me just check, by the way, um, on the phone, forgive me, um, is there anyone on the phone who wants to weigh in on sort of the end of February? We've got one conflict, but I think that conflict may not be, Michelle? Yes?

>>: I was just getting a heads-up via text here that, um, Hems is the week of February 11th to 15th in Orlando. That's another huge conference, but there might be some potentials to, um, we would have to make sure people get hotels very quickly, um, that's a huge one, but the medical folks would be there in many cases.

>>: Good to know. It also means that we can't schedule it outside of that, during that week. Um, thank you. So, we will, um, we will work on nailing down a date. I think Bruce's advice was good, which is trying to nail down a date, and then we will talk location. Um, we're also, again, trying to be mindful of getting people on planes. Would it be useful to, sometime around the second or third week of January, have a two-hour, three-hour, um, you know, phone and slide share, to give folks, um, here's something to aim

for, here's, let's try to get an intermediate draft, is there value in having that on the calendar? If you remember, we did that, we had originally planned to have one in September, and just the working groups were a little slow setting up, that was my fault, so we said, you know, let's focus on just having, circulating an update, and if we didn't want to actually exchange, we could still share documents, all of the working groups could sort of have something to show by the middle of January. Thoughts on that?

>>: Yes.

>>: Yes?

(Laughing.)

>>: What was the date of the FDA workshop?

>>: The FDA workshop is the 29th and 30th of January. All right, so, what we will do is, um, we will put our heads together, folks have further thoughts on that, we will definitely be trying to circulate something, um, a few weeks after the holidays, so that, you know, the working groups can, they can do the work up to the holidays, they can take some time off, come back, put together an intermediate document that we can share. Maybe we'll, you know, all get on the phone and talk to each other, and then in the end of February,

we will try to, um, convene in-person, and we'll, um, we'll talk, we'll get the date down, and then we can talk through what that would look like. Yes?

>>: So, I missed the last meeting, so I didn't join any working groups, I'm now at this meeting, but I haven't written down my e-mail or contact information or anything anyway, and someone put up, like, a, you know, 8-font size that I couldn't read URL that I could go to to sign-up for something, so how do I sign-up to help with something?

>>: Duncan, that is a fantastic question. I'm so glad you asked it. It is certainly not too late to join the working groups. There are sign-ups on the NTIA website, I will recirculate them. Three of them are based on, um, cert.org, or the organization formerly known as cert.org, and the Lennox Foundation is hosting another WebEx, we will circulate out that information in the next day or two, so that everyone can sort of, in addition to seeing what happened, can jump in. I'm also going to ask, um, all of the working group leads to re-send calendar invites. I apologize, for those, if this confuses your schedule, but it allows us to make sure that folks, who for some reason, signed up after there was a calendar invite sent out, it's

something that will be on the calendar. So, thank you for clarifying that. It is not too late to sign-up, there's a lot of work to be done, and the final plug I'll make, um, and we've got 10 minutes left, so there will be some chance for you to jump in, is, um, this came up earlier, are there folks that you think should be in the room? We do have some customers in-person and online, I heard that feedback. I'm going to go out and see what I can do to, um, select that group Are there other folks that you think should be engaged in this discussion that you haven't really heard from? Are there sectors where you think this could be a truly revolutionary approach, but we want their feedback as part of building this out?

>>: Is there a list that we can see which one is represented?

>>: Traditionally, we've avoided sharing that list, because when people signed up, they may not have agreed to give that information, but I'm happy to talk to it. It's more --

>>: At least the sectors.

>>: Yes. I mean, we have some presence from the IT sector, but I don't think we have an overwhelming, um, presence from the enterprise IT sector. We have

one of the larger trade associations from that sector in the room, I'm sure he'll help, and, um, certainly healthcare is very well-represented. ICS is quite interested, but we just need to bring more of them to this meeting. I know a bunch of them have signed up for mailing lists. Um, Michelle?

>>: I was just going to say that we do actually have a place in our guidance documents to try to capture all of these sectors, so if you have a list of sector-specific, but not necessarily entity-specific, um, we could use that information.

>>: Excellent. All right. Um, we've sort of hit everything we tried to on our agenda. This was, I thought, a really productive meeting. We have these in-person, because it's a chance to show up and roll up our sleeves and dive in. Our first meeting was really just saying what's going on, why are we here, and there was a lot of discussion about that. In the intervening few months, you guys have actually made a lot of progress. I think we're quite happy with that, and in addition to sort of seeing the path forward that the working groups are on, I think we did a lot of good work today of really looking at some of the big questions and figuring out how we're going to tackle

them, and so, moving forward, we're going to return to our working groups, NTIA will send out a minutes, we'll post the slides, um, and we will also, um, sooner, in the next few days, when we have it, post the video online, so, um, you know, if there's something that you want to capture from this, you can give it a re-watch, make popcorn, and then I think I'm going to sort of say is there any last words or things that folks want to say about how we should move forward? Duncan?

>>: I hate to be a pain on this, but I'm having trouble finding it. So, the only place I have to go to is the NTIA software component transparency, and I can find everything? Because none of this stuff is on that site.

>>: Thank you. Um, there is, and it used to be at the very top of the page until we put up information on this meeting, there's a link called October 1st update, and that will have some of the sign-up information, but you're exactly right, it is something that, um, we put prominently as this was standing up, and we lowered it on the website to make sure that folks could find out information on this meeting. So, penultimately, I want to thank Megan and Anna for helping to make this meeting happen, and Anna for

documenting the work. She is at Stanford and will soon be looking for work in the next year or so, so if you're looking for someone who has her fingers on the pulse of technology policy, and then the final thing I will say is there is a bar not far from here, um, the name of which I will look up once we've stopped the web recording.

(Laughing.)

>>: And for folks who want to continue the conversation in a little more collegial environment, um, we will all reconvene there. To those who have been watching or listening, thank you so much, and we will reconvene in February.