>>: We're going to, um, move on to thinking about standards and formats, and unfortunately, of the three co-chairs there, um, oh, wait, sorry, we have a quick note, as you guys come and setup. So, Bill, you get the last word on use cases.

>>: Let me take myself off mute here.

>>: Great. Thank you.

>>: Sorry about that. I couldn't make it, unfortunately. I just really want to highlight again the need for, um, you know, some positive correlation between what we get, you know, daily from CVEs against our products, where CVEs already have it built into it, really don't want to lose sight of that particular component. I know it's kind of a mess from a structural perspective, but getting us to a point where, um, we almost don't have to worry about what we get for information, and we can just automatically notify engineering teams, as well as customers of what's coming down. I don't think I heard that explicitly mentioned, but just wanted to get that in there.

>>: Thank you. It's a great point. All right, so, we're going to now shift from use cases to standards

and formats. Unfortunately, one of the, um, co-coordinators of this group, Kent Lanfield, is feeling a little under the weather, so he's going to be joining us, um, remotely to talk about one part of this presentation, but, um, we've got JC and Kate here and Kent remotely, and, um, so, take it away.

>>: Oh, mic on. Otherwise, I could project to the back row, which I can also do. So, standards and formats working group, um, we have the good luck to, um, convene around certain pre-existing capabilities as opposed to having to, you know, pull notions out of the air of what could be and then figure out how they would be implemented. So, what we're going to quickly do is do a summary of our activities, we're going to do a 5-minute overview of each of the standards that we've been examining in the working group, and then point to some of the next steps. So, the goals, right? Very concrete, very empirical, right? So, what exists today? Um, looking at how these different solutions can work together, where they are good, where they have gaps, and then to document, um, workable and actionable machine-readable formats, right, because regardless of which flavor of format you like, having things be machine-readable and consumable and open and

ingestible by third-party processes, right, that's a win, you know, whatever format that you're using, and we want to be very explicit about this is not about pick a winner, this is not about saying, oh, well, you know, like the Highlander, there can be only one, and this is the standard, right, because the developer community is in a constant state of evolution, asset managers have use cases.

There are, in this SBoM and software transparency world, use cases that are actually quite distinct, and it may be that, you know, one format best serves certain purposes, and another format best serves other purposes, so it's not necessarily about closing down, it's about mapping and understanding the capabilities that are there and where they overlap and how they harmonize, and also, to put this in a broader context, because this isn't just a United States context, right? There are proposed solutions here, but the global, um, developer community is actually, it's operating everywhere. The EU has its own discussions afoot and, you know, standards, conversations going on, and in some senses, some different approaches to things. So, again, we have to map, right? If there are differences between the way that these discussions are evolving in the United States and the way they're evolving in the European Union, we need to understand what those differences are, so that people can resolve them. Um, I'm not going to read this aloud, this is our, um, charter for standards and formats, and, um, again, this kind of goes to what Art was talking about in terms of even just locking down on the terminology of, you know, what do we mean when we say component, um, you can have that discussion for a long time, but when you have to actually instantiate it in a format, right, that's when these things become concrete, and we have the boundary objects that we can really talk around. With that, so there are two standards that we have today that we've been discussing, software ID tags, or SWID, and, um, the software package data exchange, SPDX, which comes out of the open source community and is an open source standard. Um, and so we're going to present and talk through each of those, um, now. Um, Kate, do you want to take over from here?

>>: So, right now, what we're looking at is, you know, to what extent is what we have with SWID and SPDX similar or complementary, okay, and, um, are we all talking the same language? They've both come from very different points of view and various different use

cases, and there's a lot of things in common, and there's also subtleties, and trying to figure out, okay, how to, um, understand are we talking apples to apples or not is one of the initial tasks we're working on in the group. Um, and then how can we interact, and how can we exist with both of them is something else we're exploring here, and I guess with that, I'll turn it over to cent, who's hopefully on the microphone now.

>>: Yeah, can you guys hear me okay?

>>: We can. Thanks.

>>: Okay, good. Um, so, next slide. Um, so, this became an issue, software ID became an issue in a few different communities, and as such, um, SWID was, um, the focus initially for, um, the vendor community to look at some of these areas, license management, security and configuration management, all specific use cases that needed to be addressed, and the intent behind, um, developing SWID was really to try to find a fundamental piece, um, an ingredients list, so to speak, as Josh mentioned earlier, that, um, satisfied the needs of these, um, these communities. Um, next slide. Um, software identification tag is an ISO standard, it's a simple XML structure that's used to define that ingredient set. SWID tag is basically an

XML construct, and final in most cases that describes, um, the contents of what is in the software package. It's managed and normally put on the computing device, on the end point itself, although it's not a hard requirement, it's used, there's a lot of asset management aspects that need to have it local on device. It does support both commercial and open source, although it doesn't support, um, open source to the level that SPDX does today, and that's one of the things we're sort of investigating, where the win-win is for both. SWID can be used for software tamper detection and protection. The hatch information that's contained allows for validating the contents on disk before you start to execute them.

Digital signatures are used for authoritative data, in other words, created by the vendors, and, you know, one of the big reasons this was done was to try to address the software discovery issue on our networks, trying to figure out what's on our networks is, quite often, an archeological dig. Next slide. So, current uses for SWID, um, tag data, package verification, um, and for software inventory itself, reporting what's on the machine itself, and, um, there's different types of SWID tags that, um, I didn't really go into here due to time restrictions, but there are primary and patch tags, corpus tags, as well as, um, supplemental tags, um, that allow, um, different uses of the, of SWID. Um, software integrity, validating patches, and software being installed, um, for the purpose of SBoM today, um, as well as, um, incorporating it into digital policy definition mechanisms on the end points themselves. Next slide. So, um, SWID tags, um, identify the software release, identifies the organization that created it, the release and the actual tag producer themselves, in some cases, if it's not, um, the vendor, um, provides a means to link, different types of information for, um, license, download locations, um, issues about, um, component support, those types of things. It supports software dependencies for things that require other packages, parts, or pieces, and it has, um, it describes the names and locations of files, where they're at, um, versions, cryptographic hashes, so that you can use it in others of the like. Sorry, next slide. Um, should be on the example SWID tag, I hope.

>>: Yes.

>>: This is, um, sort of a sample of a SWID tag file, and as you can see, it is a simple XML file, it

does, um, allow for, um, capabilities to identify who, um, is the creator of the file, what the corporation is, what the product itself is, where it's at and the like, types of things that you would expect in trying to identify a SWID tag, or location. Sorry. Next slide. So, um, normally, in the past, SWID's been used by the vendor community, SPDX used by the open source community. Both of them are, um, very important here, because the vendor community is incorporating a lot of open source into the products that they sell, hence the needs. Um, vendors produce SWID tags normally as, um, at build time, so in other words, this is integrated into an automated process, in the build software, so when something is created either for internal testing purposes or for release candidate or for actual production, um, SWID tags are created to indicate, um, what those are. Um, this is, when vendors produce them, they should be digitally signing them, and when they do, that creates a starting point for, um, a chain of trust, at least from that software and the different pieces and parts that are there. Um, these are different, um, from the standpoint of how vendors, um, use them. One of the things that we've found using SWID tags is that, um, support costs are reduced, because the support folks can ask the customer to run a little, tiny program that displays the SWID information, um, as to the release information, patches, hot fixes, all that stuff, and it will exactly show them what they have, so there's no digging around to figure out what's on the disk, we know immediately.

>>: So, we may, um, we'll post all these slides. Is there one thing you wanted to grab from the last few slides here?

>>: Yeah. Well, the last, well, one of the slides is sort of messed up, 14, um, hopefully, we can get that updated, but the key here is that this is a, um, a very lightweight process, very easy to get through from, um, incorporating it into, um, our build processes and from the standpoint of, um, then having that always there. The slide that was the use in software life cycle, which will get replaced when we put it up, that shows the fact that this is really sort of an as-built kind of environment, the SWID tags match what is actually put on the device itself, so if you get an initial installation, you get a SWID tag with all of the information about that version. If a product then has a, is later patched, you'd get a new SWID tag, the old one would be removed, and the new SWID

tag actually has that current information. So, this is --

>>: Excellent, and there's also, for those who are interested in learning more, we'll post these slides online. Just want to keep an eye on the clock, but there's also a Mister 8060 that provides a lot more detail on SWID tags. So, I think we're going to, is there one last thing, or we can move on? All right, um, we're going to leave your line open, Kent, for the discussion, but I just want to sort of show there's the SWID approach, and then there's, um, software package data exchange.

>>: Data exchange, yeah. So, um, software package data exchange pretty much arose organically, and it came up from the need of, um, people actually shipping software, um, in the embedded space initially, not sure how to communicate the same information and looking at the same open source packages over and over again and having no way to communicate with each other, and, so, this, um, working group formed underneath the Lennox Foundation about ten years ago now and has been basically accreting use cases and figuring out a common language to express these use cases in, and we've had fairly wide industry participation all the way along,

um, and it's being now adopted into, um, the supply chains, being specified in some of the contracts that are going on today. So, it is, um, at the heart of it, a document, just basically has information about the document, the packages, and then most of the rest of it is optional. So, you know how we were talking, we want to have a basic main minimal viable product? This does have a minimum viable product state right now and has a lot of options for expressing things in a common way, and so we're seeing it being adopted and work from that perspective. At the heart of it, we just have the version information, which is effectively what you're seeing in SWID, as well as you may want to have some more information about verification and signing and where you can find things, but you can, if you start digging into the spec, you'll find a lot more cases of being able to put relationships into play, being able to express things, like containers, express things, like patches, and the different artifacts you do find in an open source ecosystem. To create it, you need to do some tooling, you need some meta data in your sources, and the tool creates a document, and there is, everything is checksum, so you know if it's changed out from under you, and this was a requirement that we had from legal way back when, in the sense that, um, there was a lot of organic formats that were out there already, and we had to put the signing in so that people would know if things had changed.

This came about also because we had to adhere to the compliance of the open source licenses, and we've been adding in the security information over time. Right now, there is open source tooling available to support it. The SPDX project has its own tool set, but some open source tools exist for people to basically generate and consume these documents, and there are commercial options available. For probably about the last four years, Windriver has been basically putting out SPDX documents as their standard build materials for all their products already, and then BlackDuck is able to generate SPDX documents now as well, and same with source ID. So, we're using some of these commercial tools already. They do have support for SPDX built in, you have to ask for it sometimes to know the manic incantations, but it is there, and we are working with other processes and practices to make sure that this becomes part of an effective ecosystem. Um, there's a re-use initiative that's coming out of Europe, and then the open chain for expressing how to build trust between partners and then supply chain. So, these are just a little bit more details on the document, on the tools that are there. Um, FOSSology is one I tend to be using, um, and it's able to consume and produce, the key being able to consume it and basically accrete information and bring it forwards. So, that's kind of the background on the SPDX. Anyone that wants to know more, by all means, reach out to me at break. Um, and what the group is doing right now is we're starting to actually do the comparisons and taking it to the next stage of, okay, are we matching one for one? If we talk about this, are we seeing the same things, or are we getting slight different concepts? Because, um, the SWID approach is being much more used in the asset management and in the commercial side, and, so, are we talking the same concepts, or do we need to basically reconcile, make sure that we can get things to interact?

>>: Um, also, from a supply chain perspective, in terms of the generators of software, um, sometimes, you can have formats that are overlapping, but, you know, the market adoption for an open source software developer to use a kind of software publishing format may be less, whereas a software publisher may be much

more comfortable adopting the format that works for asset management, so, you know, depending on who upstream is generating the information, there can be more comfort in the open source community, which comprises about 80 percent of commercial products to use solutions that are themselves open source, as to, um, as opposed to adopting kind of proprietary or closed solutions, whereas in the publishing, software publishing industry, um, there's a lot of comfort in the adoption of formats which, um, slide right into kind of a commercial and proprietary work flow.

>>: Tell us how you can join.

>>: Yes.

>>: And I will say, it's been, um, good to sort of see this technical deep dive, and I think, um, there's been an appreciation that, you know, it's not going to be one standard rule of them all, and now, the approach is to say how do they fit together, and the most constructive way to move forward. So, thank you, and thanks to Kent on the phone. Bruce?

>>: One of the goals, I'm not clear if it's a, exactly what kind of goal it is, was that you weren't going to select one standard. The problem is softwares consist of many components, and if some of those

components decide to pick one standard, and some provide, decide to pick the other standard, there's going to be a problem there, and, um, in the case of medium size products at Oracle, where I work, we have over 300 different components in many of the products.

>>: Only?

(Laughing.)

>>: I'm just talking about, sure, it might be larger --

>>: I'm used to seeing thousands.

>>: Okay, fine, thousands, but the problem is the same, whether it's hundreds or thousands.

>>: Right.

>>: Which is if, you know, half these guys, or some percentage of these guys, large percentage of these guys use SWID, and the other, um, large percentage use SPDX, how does that get, you know, coordinated?

>>: So, this is part of the reason we're trying to map and understand how can we do translations, how can we get the information into your own databases so you can be tracking things.

>>: Okay. I didn't see that as a bullet.

>>: Actually, the last, the slide with the mapping, right, that's the kind of empirical evidence

that that is a goal and that mapping is being done. Um, on a practical basis, facts on the ground, um, there's a global community of open source software developers that is never going to be forced to use a proprietary standard. So, we live in a world where if someone could wave a magic wand and everyone could use something perfect that was only one thing, that might be good, but we have to maintain openness in the system, and, so, I think this is a recognition of that.

>>: Yeah. I think that is the big step forward for the group, is to understand how they play together, and this is, um, the, you know, basically saying where do they, where are they similar, where is there overlap, and where is there complementarity from an organizational perspective, and trying to tackle exactly the problem you're focusing on. Kent, do you have a comment on this?

>>: No, I think we're, you know, we're really on the same page, trying to find that win-win where it makes the most sense. From the standpoint of one, you know, I would love to have one standard rule of the world, not in this case, but in every case. The reality is we have lots and lots of standards that we have to deal with, and some of them are complementary, and

that's what we're sort of shooting for, and some of them are absolutely distinct and different. That's not what we want to achieve. So, from this perspective, it's really a matter of trying to see how we can harmonize both communities so that we can get the best, um, the best result. It's, um, open software, um, is a different beast, so to speak, than proprietary software, transparency for open source is much easier to deal with, although it is still very complicated, but it's much easier to deal with than the vendor products that don't allow you to access the source. Um, so, there's different issues here that have to be addressed. While we want to try to harmonize as much as possible, we're not, there is no mandate for us to come up with a one size fits all.

>>: So, I've got Duncan, and then Omar, and then back to Bruce, and then we'll move on for the moment.

>>: Hi. So, I don't know as much in this as I wish I did, but I at least get the impression that both the two things being looked at, the SWID and SPDX, are more the format of the ingredient, not the format of the list of ingredients, and I know Olas sort of got into this issue and created something called Cyclone DX, which was sort of the how you, it has to be based

on SPDX, but it's sort of here's the format for the list of ingredients, not just the format of the ingredients. Am I getting that correct?

>>: SPDX is the format of the list.

>>: Okay, the list as well?

>>: Yeah.

>>: SPDX, um, has two key elements right now. We've got a license list, where we standardized on the set about 300 licenses right now that are commonly used, and, um, that's picked up a lot of adoption already. The SPDX documents are sort of the next stage that's being focused on, which is the list of all the ingredients and how you can map relationships between documents and between packages and how can you pull all the pieces together to put out a dystroph, for instance, and, you know, you don't want to necessarily have all of that necessarily in one big document, but you want to have relationships between these documents and so forth.

>>: Okay, so SPDX is definitely a list. Is the other one, SWID, a list or an ingredient?

>>: Um--

>>: It has the capability for both.

>>: Thank you.

>>: Yeah, one comment, just to probably capture it, one of the challenges that I'm seeing, regardless of the standard, because we can translate, is the governance in pulling the information within the fields of the standard, specifically the naming and the versions, and in the commercial products, you know, of course, you know, I'm in the thousands, but, um, we consume a lot of open source, and we contribute to a lot of open source, so depending on who's actually contributing, you may actually have, even if you have one single standard, you may have different ways to represent that product, that version, and everything, and that's the biggest thing that actually will hinder the progress here. Even the commercial products that you listed there, I consume them all, you know, I get different outputs.

>>: Yeah. Um, that's part of the reason it's important to define what are valid fields and what formats are valid fields, and we've been focusing on that, I think in both projects, and, so, the definition and getting more use case, getting more examples, getting more precedent established for people to fill these things out effectively is a goal eventually.

>>: Naming is a known hard problem, and

fortunately, we have a lot of folks who spent some time thinking about it. Dave Waltermier has been participating in this discussion very constructively. Um, Bruce?

>>: Yeah, I just want to clarify my statement. It wasn't whether I care which one is selected, I expect both will be, the question is if I have a document or a piece of software that's got both inside of it, how does that get, how does that work out? And as far as I know, neither SWID nor SPDX explain how the, if the other one's in there, how you get to it.

>>: Um, SPDX will be referencing as an external reference ID to the SWID ID, and that's what we're working on focusing right now, and we can go to CVEs, you know, CWEs, etc. We're adding more security information into SPDX, but we can go to SWIDs directly right now, if we want to, and I suspect they're looking at it from the other side now too.

>>: I'm going to encourage folks who want to dive in, um, the working group information is back up here. We want to move on and make sure we got time, but Josh, quick last word.

>>: When we did our harmonization across groups, one potential useful way to look at this is you

can do the am I affected and where am I affected with a two-column field in graph. If you were to draw all the use cases we can do and want to do, some of them get unlocked more or less as we standardize better or worse, so it's not a this is a deal-breaker, it's which, in fact, we could even make a table of which of these use cases are possible without a standard, with CPE, with SPDX, you know, with a harmonized cross-referential, I just don't want us to think it's a binary thing.

>>: All right. Um, thank you. We will, um, table that for the moment. Of course, there's plenty of time to come back to that the rest of the day.