>> And now we have the Standards and Formats Group. I'll pull your slides up here. JC, thank you.
>> All right. So I'm JC Herz. I work for a company called Ion Channel. I'm representing the Standards and Formats Working Group today because Kate Stewart, my co-chair, who is at the Linux Foundation, is out of the country.

So in some ways, we had an easier job than some of the other working groups because our job was to, you know, review what exists, right? What's the art of the possible, document that and map that kind of against itself so that we have, you know, the current state of practice and that we can lead in and hook into the efforts of some of these other working groups.

So we're chartered to review the existing standards and formats. There are two that we're presenting today. We did survey the landscape of different formats that exist that are in use because software bills of materials are a many-splendored thing.

Specifically with regard to the identification of external components, you know, what is in the software that you did not write yourself, and also, commercial and open source. So there's a lot of focus on open source, as there should be, but, you know, there are also commercial libraries that are incorporated as third-party dependencies, as well.

So the way that I like to think about this is, this is third-party risk management. And that includes commercial. It includes open source. It includes dev ops. It includes, you know, what's in your legacy stuff. Not all of those use cases are equally easy, but it all falls into the same category.

And we pulled together a group and did outreach to a lot of members of the community and industry related to this kind of transparency, and that goes from software to even imbedded as well as efforts to document software heritage and -- you know, almost like the Wayback Machine for what was that code at a certain point in time, because one of the things that we recognize and that we had to become aware of, you know -- particularly, it's an area of my personal focus -- is audit. It's not sexy, but people need to be able to prove that they knew certain things at certain points in time, which I would even add, as a use case to Art, it's a little bit different than vulnerability management, right? Compliance is kind of a different creature. So how do we get near some of those requirements, because that he very much are real in regulated industries.

So these were our goals. We wanted to determine what solutions are, how they can work together. There was specifically not a requirement to find a single solution. This was not, you know, one format to rule them all, because different formats were developed to different against different use cases. And I'll say a little bit more about that in a minute.

And to document these things, you know, what are the workable and actionable machine-readable formats so that we can produce -- we have a white paper, which I believe was distributed in addition to this deck, and it's very basic. Our next deliverable is going to be more of a how-to guide. So I think we've got the "why" and the "what." Now we need the "who" and the "how."

And that more technical deeper dive with reference implementations and, hopefully, links to code in open public repositories, that's going to be our next big chunk of deliverable.

We also wanted to take care to audit the whole international landscape, because, you know, the EU -- there are discussions afoot about this. It's not just a U.S. problem, for obvious reasons. And the software industry itself is global, right? So if we come up with some special unicorn solution that only works within the confines of the U.S. and that only U.S. people care about, that's good for U.S.-only use cases, but everyone else is not going to adopt it because it doesn't speak to the needs.

So the "what." And we kind of tried to act as much as we could, like, as a development team. So we ran sprints. We have these very helpful and cheering "done" marks next to the tickets that are accomplished. And, you know, things that are of course, you know, the other group's purview as remits to us are marked "pending," as any good development team will do, to make the pending items somebody else's job.

So we did work and cross-cut to some of these other groups, particularly the use-case group. There's a bit of a chicken-egg, right? So what do you need it for? Well, you know, what does the format do? But I think we've managed to resolve this, and we're developing a taxonomy of both simple- and complex-use cases, because some of these cases are relatively simple in their definition. For instance, IP management. I think that's a relatively mature domain here. Other use cases, you know, high assurance, you know, security and, you know, vulnerability management and supplier selection, more complex and less mature.

So we're going to build these use cases out in the context of our how-to guide, which will provide that next level of granularity and depth.

And we want to get, you know, very close, if not to, actual working code for a lot of this because particularly in SPDX is open source, and there's a lot of tooling around that.

So what does success look like for the Standards and Formats Working Group is, it's got to be machine-readable. In my history, I have seen software bills of materials delivered as PDFs to customers. Not helpful.

There have to be some direct linkages so that, you know, whatever automated process is working on these things can get some notion of, you know, what points of origin were. And it all has to be automatable and verifiable. And verifiability is very important because there have also been cases that I've investigated where you have a beautiful body of evidence. You know, it's thorough, the diligence is amazing, it's granular, like, everything's in there. It just doesn't happen to match the artifact that was represented, right?

So man-in-the-middle attacks are a thing, and it's not uncommon, right, as a vector to say, Okay. You've got this beautiful body of evidence. Anything that body of evidence is next to is going to go through, right? So let's just demi with that. So we have to provide some verifiability both to the package and that the format hasn't been messed with, either. So this is very important.

Documentation of these capabilities and usefulness both in an open-source context and in a commercial context and in different phases of the software lifecycle, because consumption and deployment of software is not the same universe as development of software. And we have to acknowledge that.

And it has to be actionable in a lightweight fashion, right? So if you have to, you know, get a million dollars' worth of tooling and hire five people to implement software

bill of materials format X, it's not practicable for all but the biggest customers and the biggest suppliers, which is going to limit, again, usefulness.

And ideally, we want to look for standards and formats that can be integrated into the production of software, right, and run in the background. And this kind of goes with the sort of lightweight nature of where the technological landscape is moving towards cloud-based stuff, towards automation, micro services, et cetera, and lowering of the maintenance burden, as well as extensibility. Because as much as possible, you know, we're all looking to future-proof this stuff, because we don't want to all be having this conversation again in three years to accomplish the same thing because the old stuff is outdated.

So minimum viable. In our alignment with framing, we've done a mapping of SPDX, which is stewarded by the Linux Foundation, and SWID, which is an ISO standard, of what they have in common and what they don't have in common and mapping against the sort of minimum viable, right? So supplier is something that they share. The component, the naming, and unique identifier or sufficiently unique identifier, as Art might say.

There's version, you know, verifiability of checksum or hash, and there are relationship fields in both of these. And I don't think we had quite the MVI, so the things that were under discussion, I think, are already in the MVI. So --

>> They are.

>> -- that's great. They're discussed and resolved. There are some differences because of where these formats emerged and the uses that they serve. So SPDX very much came about in a software-development context. So there's a lot of fields and a lot of granularity around transitive licensing, right? Because in order to properly manage open-source software, you really do need the transitive licensing. Otherwise, you know, there's GPL in there somewhere, and the whole point of it is that it is inherited as well as, you know, just lots of different component identities and extension and augmentation points.

In SWID, there is a whole realm of fields and categoricals related to entitlements and procurements, because that is a context in which SWID was developed and implemented which is, you know, if you have a software publisher, is that software publisher being paid appropriately according to a contract that has been written between the supplier and the customer? And SPDX doesn't really have anything for that because it emerges at the point of origin rather than being designed for point of consumption and deployment for entitlements.

So these are how the categories are distributed. There are mappings between them in the sense that one can contain a link to another, which is really convenient. And there's certainly a universe in which both are used by the same enterprise in different use cases. Because if you're the CFO managing, you know, whether or not Adobe gets paid at the end of the month versus a dev shop doing in-house software development in the cloud, those two groups of people have different needs, and there are different formats that serve them with different sweet spots.

So again, that's sort of the mappings and the relationships. And we have made really, really great progress in the compare-contrast and the mapping between them. So we have accomplished something technical in this group.

And again, you know, for use cases, there is overlap between these formats, but there's different fit for purpose. I sort of covered that previously.

Workflows and tooling. So this is one area where, in SPDX, because it comes out of the open-source software community, there's almost too much tooling. There's open-source tooling. There's proprietary tooling. We probably need to do a better documentation job of what those tools are. That's going to be part of the forthcoming how-to guide.

With SWID, it's a little bit ambiguous. It's a proprietary standard, but you don't necessarily need the documentation in order to make use of it. However, there's sort of -- in the open-source community, there's always a cultural resistance to things that have -- you know, are proprietary and have a paywall. It doesn't matter. It could be a nickel, and it's proprietary and it's not open source.

There is available documentation. We're still trying to chase down and document the tooling to produce SWID. So, you know, there's proprietary tooling to produce it. There is some open-source tooling to produce it. You know, we still have to research the landscape for that. In fact, in open source, there are a lot of projects that don't consider themselves products. And so it's unclear whether they will produce either SPDX or a SWID tag. I think at such point that someone wants to get paid for something, it becomes desirable to use one or both.

Questions and discussions. You know, we have -- we're working with use cases now. We're working with the Framing Group, and, you know, these are the asks that we have. In the meantime, are there any questions from the audience? Duncan.

>> Hi. Duncan -- (inaudible). If you go back to your "What does success look like" slide, the slide right before it, you have a lot of "dones" and "pendings." That slide doesn't have any "dones" and "pendings." Can you do a real brief --

>> Right.

>> -- where you're at on that one?

>> So the thing is -- so for these are both machine-readable formats, so --

>> Or just sort of yes-no answers.

>> Yes.

>> From a group viewpoint, we think we've got this one.

>> Yes, absolutely.

>> We think we've got this one. This one is still --

>> So we definitely -- the formats are machine-readable. They are documented capabilities. People understand how to interpret them. The interpretation of those formats is actually in practice, right? So people are using SWID for entitlements management. They are using SPDX for, you know, supplier, you know, assertions and for vulnerability management.

The commercial world is using SWID. The open-source world isn't. But, you know, there's utility in both of these software galaxies. And the distinction between them is actually a little bit artificial at this point because open- source software is in everything, but only some people are getting paid.

>> SWID is being used in the open- source world. Red Hat is a producer of SWID tags and major contributors to open source --

>> And I think that's almost the exception that proves the rule, right? Because Red Hat's getting paid for its open-source software. And, you know -- so, yeah. But I think the

commercialization of the management around the software, it's commercial packaging of open-source software. And at the end of the month, Red Hat needs to get paid, too, right? So why wouldn't you have a SWID tag.
>> I think the takeaway is not who is doing what. It's that --
>> Yeah.
>> -- we can live in a bilingual world, and folks can sort of pick whichever one they want.
>> Yeah. So actionable in that software publishers can implement in a lightweight fashion. We still need published reference implementation for SWID; or if it does exist, we need to find it and tag it. So that one is pending.

In terms of being an integrated part of the development process, there are a lot of implementations of people using, you know, all kinds of software-composition analysis, SBOM generators. You know, we furnish one. There's a whole bunch of other companies. GitHub does, to some degree, as well.

I think, at this point, it's in the thousand-flowers-blooming stage where everyone has cobbled together something if they're asked to, you know, particularly in financial services. But there's not yet a lot of convergence around the design patterns that work. Because there may be particular ways of arranging the duct tape and baling wire that work, or, you know, there are fully kitted-out -- you know, if you want to go and buy Veracode and get a huge, heavy IDE, right -- I mean, if you're into some of these heavier solutions that -- you know, like CAST or like multimillion-dollar enterprise implementations that take months, a lot of them have some beautiful functionality.

I think the question is -- and we probably need to make a graph about this -- is, aside from those heavyweight solutions that kind of do it all but simultaneously lock you in, are there lightweight ways to achieve the same things? And this particularly becomes important when you have the suppliers of important capabilities not being well-capitalized organizations, right? There are elite software developers, you know, and it's a 20-man shop, and they're being used in national security, right? So how are those guys going to, you know, in a lightweight fashion, make automation of their SBOM and, you know, for continuous authorization, something that works in the background.

That's kind of a market problem as well as a technological problem. But we're very confident, you know, because we and other people are delivering this stuff, that it can work.
>> And I think that's something we're going to talk about a little bit more this afternoon is how do we want to think through the tooling side of it, what does outreach look like for this community, and how do we make these resources available and, you know, how can we split that into the work that we're doing along different working groups. Because I think that's come up in almost every discussion we've heard.
>> Oh.
>> Can you speak to "verifiable to the package it represents" with a little bit more detail as to --
>> Okay. So there's different -- there's different ways to think about that. One is, if you have a -- if the package has a hash, right? Everyone talks about hashes. You know, can you say that this thing matches what's back there?

In ecosystems that produce hashes, that can sometimes be useful. I believe that hashes are actually overtalked about because very few organizations have the capability to actually -- or the willingness to actually check them, number one.

Two is that the way that hashes themselves are produced is very problematic. So it's not -- there could be a digital signature from a supplier, right, which is one kind of hash; but sometimes people are talking about these hashes that are produced in the compiling of a piece of software.

And the problem with that is that, depending on the compiler, date and time might be incorporated in the hash- generation process. So if you compile it, you know, one second apart, it's going to be a different hash. So it's not very useful.

So at Ion Channel, what we do -- and this is just one method -- is, we'll crypto-couple the assurance to the payload. So we'll take a machine-readable piece of evidence, and we will take the hash of the artifact, and we will put the hash of the artifact that we have just analyzed as part of the analysis process into that evidence and then deliver them both at the same time.

So essentially, they're coupled together cryptographically so that if you change that package, it would no longer match what was in the evidence. So there's point of origin. There's also, Okay. Wherever I got it, is the evidence actually about this thing that I just analyzed and keeping those things together? Because it is digital, but you don't necessarily want to trust copies.

And then thirdly, there's the verifiability of the format itself, right? So you can do an implementation of -- we just did an implementation of SPDX in NIEM-compliant XML, you know, just as an exercise. And that means that you can check that the format actually is what the format is supposed to be so that fields haven't been stripped or that, you know, there aren't fields in there that are not supposed to be in there.

So essentially, there's a master template for what the format's supposed to be that the person who is the end user or some automated process that the end user can always check to say, you know, Is this format actually valid?

And that becomes useful in high- assurance environments, right? And, you know, it's not necessary in every environment, but if you're in a high-assurance environment, you got to start doing stuff like that.

>> That leads to my last question. So what does success look like for that -- for the verifiable context?

>> So in the verifiable context, it means you have a verifiable format. So for SPDX, if you have, like, a NIEM-compliant implementation of it, there's a master format sitting up there in the universe at an authoritative place that you can ping and say, This matches. This is a version of that.

I believe, you know, SWID is an ISO standard, so you could just have the standard and check, Is this the right standard?

The verifiability of the evidence, the coupling between the evidence and the payload, right, that's a method. You know, we haven't seen it in a lot of other SBOMs, but you can certainly do it. You can do it as an augmentation point for SPDX.

And then the verifiability of, you know, sort of point of origin is kind of a step beyond that.

But I think -- and we can discuss this in the afternoon -- depending on how high assurance you want to get, you might just want to take flat code, you know, through an assured-build process because you don't trust the internet to build your software for you. It really just depends on what the consequentiality is of your particular software production and consumption.

>> Thank you.
>> And the final thing I will add into the great work that the Standards and Formats Group has done is they have a very solid draft white paper. I'm going to make a small quibble with the full bold "done," because what I don't think it has is, I think this group has not taken a big stick to it.

And so we have posted it. I sent it out earlier this week. It's on the NTIA website. And when I send out a follow-up, I will send people another pointer to it. This is exactly --
>> But it's perpetual beta --

(Laughing)
>> -- so user feedback is --
>> Yes. So I think we want to make sure that there can be some solid feedback on the white paper, because I think it is great. It represents a lot of work by a bunch of people, particularly the folks who have been leading it. So thank you.
>> Okay.
>> So, Allan, is the feedback mechanism emailed to you, document in a G Doc, attend the meeting? If we want the feedback, what's the mechanism?
>> That is a great question. JC, do you have a desired view on what is the best way to get feedback?
>> I think that feedback could either be directly to me and Kate; or if you're on the Standards and Formats email list, that would be great. If you want to get on the email list, request that. But, yeah, if you want to -- Allan, if you want to be a relay --
>> Sure. Happy to.
>> -- if anybody has -- you know, we'll catch whatever you forward on to us.
>> Excellent. Thank you.
(Applause)