Plugfest FAQ

Q: How come I wasn’t invited to the plugfest?

A: Sorry. The organizers circulated a form soliciting interest. All respondents expressing interest were given the opportunity to provide ‘sweat equity’ by either producing or consuming an SBOM. All qualifying entrants are being invited to the plugfest.

Q: Is this a bake-off to find the best tool?

A: No, the goal is very much not to compare tools, products, or companies to rank them or in any way recommend them. Instead, it is to help align tools and data formats to maximize interoperability.

Q: What are the expected outcomes of this plugfest?

A: The organizers are working to create the following outcomes for the broader SBOM community:

* To create set of "reference" basic SBOMs that any tool can use to check themselves.
* To validate the interoperability of the "baseline" implementation of SBOM in the 3 consensus SBOM formats (SWID, SPDX, CycloneDX).
* To gain insight into what work is needed for future plugfests tackling more complex software and help the SBOM community.

Q: For the blinky.ex Nerves example, what should we build the SBOM of?

A: Given the nature of the embedded systems for Nerves, we will accept SBOMs of either the blinky.ex file or the blinky folder within the nerves-project.

Q: Why isn’t there a python source code example?

A: None of the organizers volunteered a python source example and there are only so many hours in a day. If you are interested, you may want to submit an example in your directory and it may be considered for the next plugfest.

Q: Why isn’t there a java source code example?

A: None of the organizers volunteered a java source example and there are only so many hours in a day. If you are interested, you may want to submit an example in your directory and it may be considered for the next plugfest.

Q: My real life software is more complicated (many more files, multiple programming languages, mix of open-source/proprietary, …). Why are these examples so simple?

A: This is the first of hopefully many plugfests and the organizers adopted a crawl/walk/run model - and we need to crawl before doing more advanced use cases. If you are interested, you may want to in your directory submit an example of what you’d like to see in a future plugfest and the group can consider it.

Q: In SPDX, how do I denote “known vs known/unknown” in the dependencies?

A: In cases where there are "known unknowns", the use of the keyword NOASSERTION can be used on the right hand side of a relationship to indicate that the author is not asserting whether there are other SPDX elements (package/file/snippet) are connected by relationships or not. ie. There could be some, but the author is not asserting one way or another.

Similarly, the use of the keywords NONE can be used to indicate that an SPDX element (package/file/snippet) has no other elements connected by some relationship to it.

The use of NOASSERTION or NONE is not mandatory for any relationship from an element. If no relationship of a particular type is specified, then the document author is not presumed to be asserting whether or not there are relationships of that type. If some relationships of a particular type are specified, then the document author is not presumed to be asserting whether there are more possible relationships of that type.

Q: In CycloneDx, how do I denote “known vs known/unknown” in the dependencies?

A: somebody fill in the answer

Q: In SWID, how do I denote “known vs known/unknown” in the dependencies?

A: somebody fill in the answer