***Ion Channel Response to Area 5 of NIST & Executive Order Executive Order 14028 of May 12, 2021: Improving the Nation's Cybersecurity***

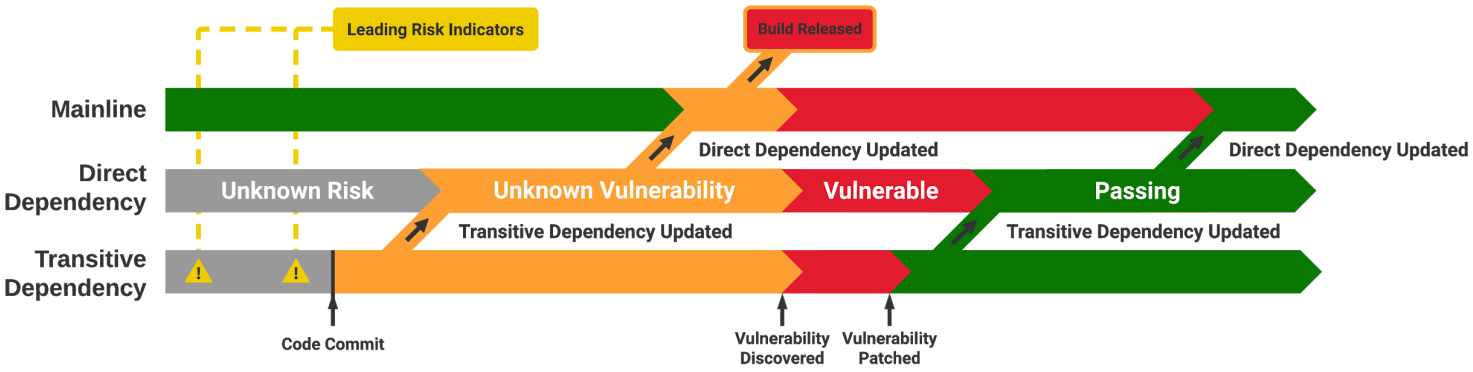
This white paper is a response to **Area 5** of NIST's call for responses to Executive Order 14028: ***guidelines for software integrity chains and provenance***, referencing EO Sections 4(e)(ii, vi, and viii), regarding demonstration of conformance to secure software development environments and the maintenance of accurate and up-to-date data, provenance (i.e., origin) of software code or components, and controls on internal and third-party software components, tools, and services present in software development process, and performing audits and enforcement of these controls on a recurring basis.

One of the biggest myths in buzzword-laden world of technology tools is that "SecDevOps" is sufficient to ensure continuity of assurance. While it's better to run software composition analysis and software testing as part of a build pipeline than after the fact, the reality is that most software is not built every day. Most software, especially software in critical infrastructure, is built frequently in its initial phases, through to product release, after which the build cadence slows down. Tight coupling between the cadence of build and assurance is viable in high-cadence domains like advertising and e-commerce. But in regulated industry and federal contracting, where capabilities in sustainment might not be worked on for weeks, tight coupling between development and assurance creates dangerous discontinuities: how much time elapses between emergence and detection of a risk by a supplier, even one running pipeline SAST on a lightly maintained capability? How much time does it take them to remediate the risk, and then how long does it take them to ship a remediated update?

Several measures can mitigate these discontinuities:

1) Suppliers provide auditable evidence that they are analyzing their product on a scheduled basis, whether or not that capability is actively being developed
2) Data and tools for testing and analysis are updated on a high cadence, so that today's test can fail yesterday's passing capability based on new information.
3) Time-to-remediation is logged by testing and analysis tools and processes within the software development life cycle, either directly or via export of test data and ingestion/processing by configuration management and other systems of record.
4) Suppliers should be made aware that customers are conducting similar analyses, either of software products directly or based on Software Bills of Materials, and that a supplier's inattention to their duty of care in maintaining continuity of assurance will not leave their customers in the blind.
5) Software that is deemed critical should also be compiled from source: this is critical to eliminate attacks impacting 3rd party repositories that hold pre-compiled binaries and have been proven to be vulnerable to attack and spoofing.

6) Where possible, both suppliers and customers should monitor for leading risk indicators, not just lagging indicators of risk such as a CVE. See figure below.




Another discontinuity that exists in SecDevOps is the information gap between the dependency mapping of vulnerability databases such as the NVD, which leverages the source code graph of source code dependencies, and the actual composition of binary packages in package managers, which differ from source code components in security-relevant ways. For example, one component from a medical device analyzed by Ion Channel has no vulnerabilities in an NVD lookup - and if you check the source code for that component, there are no known vulnerabilities. But if you download that package from a package manager, as most developers do, there is a highly vulnerable component incorporated into the binary as a runtime dependency. The NVD doesn't understand that the package is vulnerable because the vulnerable component isn't "in" the named component. It's "with" the named component as packing material. In order to accurately account for software provenance, point of origin is required to disambiguate the security profile of a source code component from a binary package which may contain vulnerable inclusions as runtime dependencies that don't flag for CVEs. In the absence of point-of-origin data, both the supplier and the customer must assume that components are binary packages and account for the presence of runtime in those package. A presentation on this topic will include sample data to illustrate risk discrepancies and discontinuities referred to above.

**Submitted by:**

***JC Herz (female presenter):*** . Co-chair, Department of Commerce (NTIA) multi-stakeholder working group on Software Transparency and Visiting Fellow at the National Security Institute, GMU Scalia Law School.

**References:**

JC Herz (co-author), OWASP Software Component Verification Standard, https://owasp-scvs.gitbook.io/scvs/ - jc.herz@ionchannel.io, 202.213.3151, www.ionchannel.io

JC Herz, "Maintenance: The Biggest Risks are Boring," internally funded case study: https://ionchannel.io/media/Ion%20Channel%20-%20Maintenance%20and%20Risk%20Management.v.2.pdf.