PUBLIC SUBMISSION

As of: June 19, 2021
Received: June 17, 2021
Status: Pending_Post
Tracking No. kq1-lkem-p9ik
Comments Due: June 17, 2021
Submission Type: Web

Docket: NTIA-2021-0001
Software Bill of Materials Elements and Considerations

Comment On: NTIA-2021-0001-0001
Software Bill of Materials Elements and Considerations

Document: NTIA-2021-0001-DRAFT-0018
Comment on FR Doc # 2021-11592

Submitter Information

Name: Anonymous Anonymous
Email: phil.shea@datadoghq.com

General Comment

As a leading Software as a Service (SaaS) company that significantly depends on third-party software, there is keen interest in ensuring appropriate implementation of SBOMs in the industry. We propose a system that resolves many of the questions posed in the Request For Comment (RFC). This will briefly describe the system, followed by how it resolves these questions.

We propose using the following methodologies and associated technologies, all of which have been recommended by The Linux Foundation (https://www.linuxfoundation.org/blog/how-lf-communities-enable-security-measures-required-by-the-us-executive-order-on-cybersecurity/):
1. Define SBOMs in a standardized manner that will ensure interoperability, using existing technologies such as SPDX (https://spdx.dev/).
2. Open metadata standard that provides the ability to record signed attestations about the SBOM. in-toto is the preferred standard (https://in-toto.io/) for assurance of the content of signed SBOMs.
3. A compromise-resistant transport protocol for SBOM and attestation data should be included in the standard. The Update Framework (https://theupdateframework.io/) and Uptane (https://uptane.github.io/) are preferred, existing methods of ensuring packages and their SBOMs have not been tampered with.
4. The proposal that the “ideal SBOM should track dependencies, dependencies of those dependencies, and so on down to the complete graph of the assembled software” is potentially unwieldy. Rather, storage of SBOMs in a public, immutable ledger that can be referenced in dependent SBOMs and queried by automated tooling is an option that should be pursued. sigstore (https://sigstore.dev/) can be employed to provide a permanent, immutable record of the provenance of all SBOMs.

Why use these technologies?
1. Software-as-a-Service and online services: even if SaaS providers do not install software on end-user machines, they can nevertheless internally record permanent proofs about their software supply chains. For example, providers may use CNAB Security (https://github.com/cnabio/cnab-spec/blob/main/300-CNAB-security.md), which employs TUF and in-toto. This is useful to validate web applications during assessments and audits. SaaS providers have valid concerns with publication of all of their internal system

dependencies potentially being publicly available, but the use of these technologies would allow a cryptographic fingerprint to be published that could be used to verify a privately-held version of the SBOM of the service.
2. Threat model: tools (notably in-toto and TUF) were designed against nation-state attackers in the first place, and have a proven track record in security (https://theupdateframework.io/audits/).
3. Authenticity and integrity: Inherent cryptographic guarantees.
4. High-assurance use cases: tools (notably in-toto) are flexible to cover additional information for such use cases, including vulnerability scan data.
5. Delivery: tools work over existing transport mechanisms, and may possibly be deployed even in airgapped environments.
6. Depth: together, these technologies allow for recursively querying the entire graph of dependencies of any software artifact.
7. Vulnerabilities and risk management: tools (notably SPDX and in-toto) allow for recording information such as dependencies and vulnerability scans that allow end-users, including machines, to make dynamic decisions depending on their risk tolerance.
8. Open source: tools are open source, and as such free of patents and other intellectual property rights that encumber them from being standardized or deployed widely.

These technologies are production-ready today. We have successfully deployed in-toto and TUF in what is among the industry's first publicly-verifiable, compromise-resilient software delivery system. Using the methodologies and technologies described above, we are able to detect a supply chain man-in-the-middle (MitM) attack anywhere between our developers and end-users of our published software.