in-toto Due Diligence for CNCF Incubation

October 2019

Primary Author: Santiago Torres-Arias
Reviewers and other Contributors: Justin Cappos, Lukas Pühringer

Background

Request for Incubation

in-toto CNCF presentation

in-toto, Latin for “as a whole,'” is a framework that holistically enforces the integrity of a software supply chain by gathering cryptographically verifiable information about the chain itself.

Modern software is built through a complex series of steps called a *software supply chain*. These steps are performed as the software is written, tested, built, packaged, localized, obfuscated, optimized, and distributed, among other steps. In a typical software supply chain, these steps are “chained” together to transform (e.g., compilation) or verify the state (e.g., the code quality) of the project in order to drive it into a *delivered product*, i.e., the finished software that will be installed on a device. Usually, the software supply chain starts with the inclusion of code and other assets (icons, documentation, etc.) in a version control system The software supply chain ends with the creation, testing and distribution of a delivered product.

Securing the supply chain is crucial to the overall security of a software product. An attacker who is able to control any step in this chain may be able to modify its output for malicious reasons that can range from introducing backdoors in the source code to including vulnerable libraries in the delivered product. Hence, attacks on the software supply chain are an impactful mechanism for an attacker to affect many users at once. Moreover, attacks against steps of the software supply chain are difficult to identify, as they misuse processes that are normally trusted.

Currently, supply chain security strategies are limited to securing each individual step within it. For example, Git commit signing controls which developers can modify a repository, reproducible builds enables multiple parties to build software from source and verify they received the same result, and there are a myriad of security systems that protect software delivery. These building blocks help to secure an individual step in the process.

Although the security of each individual step is critical, such efforts can be undone if attackers can modify the output of a step before it is fed to the next one in the chain. These piecemeal measures by themselves can not stop malicious actors because there is no mechanism to verify that:

1) the correct steps were followed, and;
2) that tampering did not occur in between steps.

in-toto is protects the software supply chain *as a whole*. In order to achieve this, in-toto provides a series of mechanisms to define:

- What steps are to be carried in the supply chain
- Who can carry out each step
- How the artifacts between each step interconnect with each other

This way, in-toto can allow *project owners* to define (and update) the topology of the supply chain in a file called a *software supply chain layout* (or just layout for short). In addition to this, in-toto provides a way for *functionaries* to provide cryptographically-signed attestations (or *link* metadata) that can be used to verify that all steps within the supply chain were carried out according to the specification.

Alignment with Cloud Native

One of the most pressing security problems in cloud native is the secure delivery of container images, as well as the verification behind their development practices. In addition, strong development compliance is a pressing matter for high-assurance scenarios. in-toto addresses this issue by providing a secure and trustworthy means for representing all the operations within the cloud-native pipeline and verify that they were carried out to the letter.

Incubation State Requirements

1. Adoption

in-toto is being used by various customers across the spectrum. From the Debian project for its reproducible-builds initiative, to cloud security vendors such as control plane and observability companies such as datadog. Further, banks serving thousands of companies and to secure the pipelines of hundreds of internal teams. Through these integrations, companies such as Twitter, Dreamworks, NASDAQ, Activision, The Washington Post, Comcast, Deloitte and FedEx and thousands of others have benefitted from the security benefits that in-toto brings to the table.

Plenty of adoption is also forecasted. Teams in Microsoft have showed interest in integrating in-toto for their internal products. Other teams such as the Cloud Native Application Bundle (CNAB) team have included in-toto as part of their security specification. Contributions from GitHub have helped in-toto update its specification to accommodate URI-like artifact references as encoded in an in-toto Enhancement (ITE-4). Google's Grafeas intends on supporting in-toto as a first-class element in their specification, and the SPDX specification will include in-toto links as a way to secure the integrity, pedigree and provenance properties of their software-bill-of-materials document.

For the sake of fulfilling the requirements for incubation, the following deployments are listed:

- Datadog
- Control Plane
- Debian

More detail about adoption, including some of the in progress adoptions, can be found on the in-toto website.

2. Committers and contributions

As an intentionally minimal security specification / framework, we deliberately do not have a high degree of feature additions in the project. Effort comes on either the implementations, such as the golang implementation (used by the K8S admission controller), the Python reference implementation (used by Datadog), the Java implementation (used by the Jenkins plugin and Rabobank), and the specification (where Datadog, Microsoft, the Linux Foundation, Google and Github can ensure metadata is inter-operational).

- Python reference implementation / specification (8 committers, 5 organizations)
    - Santiago Torres-Arias (NYU), Justin Cappos (NYU), Trishank Karthik Kuppusamy (Datadog / NYU), Lukas Puehringer (NYU), Vladimir Diaz, Sebastien Awwad (Conda), Ofek Lev (Datadog), Reza Curtmola (NJIT), Holger Levsen (Debian)
- Golang implementation (3 committers, 2 organizations)
    - Santiago Torres-Arias (NYU), Lukas Puehringer (NYU), Luke Bond (Control Plane).
- Java Implementation (4 committers, 3 organizations)
    - Santiago Torres-Arias (NYU), Lukas Puehringer (NYU), Mojtaba Zaheri (NJIT), Gerard Borst (Rabobank.nl, not merged to master)

We have had active contributions from an array of contributors across the CNCF landscape. One way to see this is via the substantial changes that made their way into the specification.

Changes to the in-toto standard largely come in the form of ITEs (in-toto Enhancements). There is a substantial ITE, ITE-4, that standardizes non-file artifact specifications for in-toto metadata. The stakeholders in it are Github, Conda, SPDX and Google's Grafeas.

3. Demonstrate ongoing flow of commits and contributions

Throughout the projects, there has been substantial participation from multiple communities. Since in-toto was admitted into sandbox there have been the following contributions:

- In-toto was officially admitted as a debian package in the debian repositories.
- Two releases, one minor and one patch releases have been published
- A new ITE draft was proposed by the community
- Rabobank proposed a refactor of the Java implementation