in-toto Due Diligence for CNCF Graduation

August 2023

Primary Authors: Justin Cappos (NYU), Santiago Torres-Arias (Purdue University)
Reviewers and other Contributors: Lukas Pühringer (NYU), Aditya Sirish A Yelgundhalli (NYU), Trishank Karthik Kuppusamy (Datadog)

This Document

The purpose of this document is to provide the in-toto project technical review Due Diligence as described here. The overarching goal of this document is to provide the requisite information for in-toto's graduation.

Background

- Link to TOC PR
- Link to presentation (TBD)
- Link to github project

Project Introduction

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 software supply chain. This information can be checked against a policy which enforces that certain steps were performed by different parties, those steps had certain outcomes, and that the result of a step was used by other steps to make the software.

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 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 that can be used to verify that all steps within the supply chain were carried out according to the specification.

Project Provenance and Collaborations

This project was started by the research teams of Prof Santiago Torres-Arias (formerly a PhD student at NYU, now a professor at Purdue), Prof Justin Cappos (NYU), and Prof Reza Curtmola (NJIT). It was originally funded with a small DARPA (1M USD) grant and has two NSF grants (totaling about 2M USD) currently. We have applied for more such funding to continue to work on the project at high velocity.

Note however, that a lack of funding does not mean a lack of support or effort for the project. For example, the TUF project (which in-toto maintainer Justin Cappos leads and Santiago

Torres-Arias participated in), has had funding for only a few years of its lifetime (about 4 years and 772K USD direct funding over its 13 years). Academics can usually find a way to work on topics of interest, even without funding.

Furthermore, the in-toto project is well supported and widely used by industry. This is through three major axes: as a standalone project, as the default mechanism to provide SLSA attestations, and within the Sigstore ecosystem.

As a standalone tool, in-toto has been used by Datadog and the reproducible builds project for more than three years. Recently, in-toto became adopted by the npm cli as part of its attestation mechanism. Other emerging projects such as SCAI (part of Intel Labs), Keylime and the GUAC project all support in-toto. Similarly, static analysis tools such as Aqua's trivy can consume in-toto-signed SBOM's for validation, as well as generate attestations of scanning.

Through SLSA, in-toto has seen widespread adoption by major CI players. This includes Tekton through its chains project that generates in-toto SLSA attestations as well as its own specific payload. Jenkins hosts a plugin to also generate in-toto attestations with minimal setup. More recently, GitLab added support for generating in-toto attestations of runner runs. Lastly, GitHub now generates a collection of attestations (including SLSA) for NPM builds as part of a public beta.

Sigstore is a project that initially started as a "transparent transport for in-toto" as originally written by Luke HInds. While its goals and scope have grown, in-toto is still one of the most prevalent types of data generated. Sigstore supports generation and validation of in-toto attestations through its cosign tool as well as its policy controller. In fact, in-toto attestations are the second most popular type of signature hosted in the public-good instance of the sigstore project.

We document broader integrations in our friends repository.

Alignment with Cloud Native / Need

One of the most pressing security problems in cloud native is software supply chain security. in-toto addresses this issue by providing a secure and trustworthy means for representing all the operations within the cloud-native pipeline and verifying that they were carried out to the letter.

A good way to understand the need for in-toto in the Cloud Native space is to understand the value of signed SBOMs vs in-toto metadata + layouts. A signed SBOM indicates that some party (whose key you presumably have a reason to trust) states what the software contains. In contrast, in-toto will have signed information about the individual steps of the supply chain cryptographically linking metadata together from various parties and validating this all against the software's policies. As a

result, their protection modes would work quite differently in many cases. For example, see the following table:

Attack scenario	**Signed SBOM Result**	**In-toto layout + metadata result**
Software manipulated after software supply chain completed	Detect and reject the malicious software	Detect and reject the malicious software
Attacker compromises VCS and inserts malicious (unsigned) code where signatures are required	Undetected. User compromised.	Detect and reject the malicious software
Attacker substitutes a malicious dependency (not signed by that dependeny's maintainer)	Undetected. User compromised.	Detect and reject the malicious software
Attacker provides files to the build system which did not come from the VCS	Undetected. User compromised	Detect and reject the malicious software
Attacker containerizes / packages binaries other than the ones the build system built	Undetected. User compromised	Detect and reject the malicious software
Tests are not run on the software but it is (accidentally?) released to production	Undetected. User compromised	Detect and reject the malicious software
The legal team has not reviewed source code licenses for included libraries	Undetected. Impact varies	Detect and reject the software

One important thing to note about the table above is that it isn't impossible for someone to do many of these steps and checks before signing an SBOM. If you did all of these checks, and signed the statements saying you did them to provide stronger validation, and distributed the root of trust for your signatures in a secure way, and managed situations where signing keys need to be revoked / rotated / expired, and handled trust delegation to different parties, and linked metadata between steps together, and let people write policies to reason about those steps, and let them link metadata in from dependencies to do so, and handled all of the above in scenarios where insiders can be maliciously interfering with your, system, then you would effectively reconstruct in-toto.

We are aware of some efforts, like the Zephyr project, where project members have worked to try to reconstruct some of the guarantees of in-toto and decided to live with the gaps in their security for other portions. For groups that have done this work already this does make sense to us as a viable alternative in the short term. However, we do believe that using a common, holistic approach like

in-toto will be necessary as projects continually add the missing security pieces from in-toto and want to reason more and more about each other as dependencies.

Note that in-toto is not a substitute for having appropriately secure steps in the software supply chain. For example, if you use an insecure process of building software that just `curl`s and builds software from a website, in-toto will happily sign metadata indicating that you did the same insecure action indicated you would.

This is why projects like SLSA and FRSCA are built as an opinionated set of steps on top of in-toto. They specify which actions they feel are more important for software supply chain security and mandate their use.

These projects are solving different problems at different levels. In-toto allows you to capture information about your steps, ensure policies about them are applied, handle trust of keys, etc. Frameworks like SLSA and FRSCA use in-toto as a mechanism to capture and enforce a specific set of policies that result in more secure supply chains.

Graduation State Requirements

1 Have committers from at least two organizations.

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 Go implementation (used by tools like Trivy and Tekton), the Python reference implementation (used by Datadog), the Java implementation (used by the Jenkins plugin and Rabobank), and the specification (where all implementations coordinate for interoperability).

Since reaching the incubation stage, the in-toto project has switched its governance model to use a steering committee. The first in-toto Steering Committee (ITSC) was voted on by the in-toto community and comprises five members from organizations spanning industry and academia. The ITSC has oversight over all in-toto sub-projects such as the specification, the Attestation Framework, and implementations maintained by the community written in Python, Go, Java, and Rust. Each sub-project has its own set of maintainers recorded in a CODEOWNERS or MAINTAINERS file in its repository. Across our sub-projects, we have contributors from a diverse set of organizations like **Google, Kusari, New York University, Purdue University, Verizon, Intel, and TestifySec**.

The current ITSC comprises of the following

- Santiago Torres-Arias (Purdue University)
- Justin Cappos (New York University)
- Jack Kelly (Control Plane)

- Cole Kennedy (TestifySec)
- Trishank Karthik Kuppusamy (Datadog)

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 standardized non-file artifact specifications for in-toto metadata. The stakeholders in it are Github, Conda, SPDX and Google's Grafeas.

Another significant ITE is ITE-6 [https://github.com/in-toto/ITE/blob/master/ITE/6/README.adoc]. This enhancement introduced the in-toto Attestation Framework to record and disseminate software supply chain specific information like build provenance, code review results, test results, SBOMs, vulnerability scans, and more. in-toto attestations are now used by GitHub for NPM build provenance, OpenVEX, Docker buildx, scanners like Trivy that can generate signed SBOMs, Tekton Chains, Sigstore, GUAC, Witness, and Archivista. SolarWinds, in their next generated build system introduced after SUNBURST, also generate in-toto attestations.

2 Have achieved and maintained a Core Infrastructure Initiative Best Practices Badge.

The in-toto project has a Gold CII (now OpenSSF) Best Practices Badge. As of 31st of July, 2023, there are only 23 projects in the world to have such a distinction. The only other CNCF project with a Gold Badge is the TUF project (a graduated security project).

According to the OpenSSF Best Practices website, the in-toto project received its initial OpenSSF Best Practices badge on January 5th, 2018.

3 Have completed an independent and third party security audit with results published

See the in-toto Security Audit ‘23.

4 Explicitly define a project governance and committer process.

The committer process should cover the full committer lifecycle including onboarding and offboarding or emeritus criteria. This preferably is laid out in a GOVERNANCE.md file and references an OWNERS.md file showing the current and emeritus committers.

The project's GOVERNANCE.md and contributor instructions cover the committer lifecycle, onboarding, offboarding, and emeritus criteria. Any participant may commit, so long as their code is approved by a project maintainer for the implementation for that codebase. The current maintainers are also listed in the repository.

The in-toto specification has a separate process by which changes and additions are proposed and vetted. This is through the ITE (in-toto enhancement) process, which involves a public proposal of a specification change which is discussed by the community.

Explicitly define the criteria, process and offboarding or emeritus conditions for project maintainers; or those who may interact with the CNCF on behalf of the project. The list of maintainers should preferably be stored in a MAINTAINERS.md file and is audited at a minimum of an annual cadence.

As above, the project's GOVERNANCE.md covers the criteria for maintainers including the onboarding, offboarding, and emeritus criteria. These are audited at an annual cadence by the in-toto steering committee, as is described in the project's GOVERNANCE document.

While the maintainers of both the in-toto specification and its implementations work together for the health of the project, for the purposes of CNCF interactions, the in-toto maintainers are the ones who will interact with the CNCF on behalf of the project. The current maintainers are listed in the repository.

Have a public list of project adopters for at least the primary repo (e.g., ADOPTERS.md or logos on the project website). For a specification, have a list of adopters for the implementation(s) of the spec.

There is a public "friends list" of project adopters.

in-toto Community Activity Inside and Outside the CNCF

We would also like to stress that in-toto project maintainers have been active members in the CNCF community.

Justin Cappos is a tech lead for TAG Security. He has been very active in the security assessment process for the CNCF. He is currently working with Ragashree Shekar to write a security assessments book for TAG-Security to help to ease the on-boarding of TAG-Security members. He is also the consensus builder for the CNCF project TUF.

Santiago Torres-Arias created the CNCF's Catalog of Supply Chain Compromises while working on in-toto and has donated it to the CNCF and TAG-Security.

We have also presented in-toto to the CNCF TOC previously as well as to TAG-Security.

We have been active in talking about in-toto, which has helped to drive adoption. Academically, this includes a talk on the peer-reviewed academic paper describing in-toto at USENIX Security 2019. We have also done significant outreach to the government and other agencies.For example, in-toto was mentioned in Microsoft's president Brad Smith's testimony to the U.S. Congress. More recently, we have submitted a white paper about in-toto to the NIST's call for position papers that was issued as a result of the 2021 Presidential Executive Order on ``Improving the Nation's Cybersecurity''. We have spoken both to Senator Wyden's office and also the staffers on the House Committee on Science, Space, and Technology that are working on a response to the SolarWinds hack and supply chain security more broadly.

In the last year, we've also worked to expand in-toto to critical industries such as automotive and other embedded software. We leveraged our experience working on Uptane, a derivative of TUF for automotive and internet-of-things (IoT) software, to understand how in-toto can be used to secure those software supply chains. We published a whitepaper entitled Scudo (https://uptane.github.io/papers/scudo-whitepaper.pdf) detailing these efforts. We've also been working closely with Toradex, an IoT hardware and software producer to deploy in-toto in their supply chain. Once this deployment is complete, Toradex's products that are used in key applications such as medical devices will benefit from in-toto's software supply chain security features.

In-toto was the first project to go through the TAG-Security assessment process. As the assessment process was being tested and refined at that time, this was much more burdensome than other projects that went through the process later.

Incubation Project recommendations

Verify in-toto's supply chain with in-toto in-toto/issue#278

See An example of this in our tooling is: https://github.com/in-toto/witness/actions/runs/8147511831/attempts/1#summary-22268394102

Improve introductory documentation to clearly communicate security scope docs/issue#15

We have worked on this process in our community repository . We also have put in a request to the CNCF documentation experts to give us an outsider's perspective.

Additional integrations, examples and/or documented case studies (such as: in-toto/issue#284, roadmap#3)

This documentation was added to our project adopters page.

Consider encoding best practices in default implementation (such as issue#287)

We've worked to address issues (such as issue#287) to reduce user confusion. We have also worked with communities that are building policies on in-toto (such as SLSA) to ensure that they have integrated it in a secure manner.

Proceed with CII silver badge & roadmap.

We have since obtained silver and gold OpenSSF Best Practices Badges

Additional recommendations

Formal security audit: no blocking issues for a formal code audit

A security audit was performed by X41, the results of which we discuss here. The issue marked "high severity" that is listed in the report was well known to us (with an issue open for several years on our issue tracker). Most other findings were related to a user potentially misunderstanding the scope of protections provided by in-toto or possibly

misconfiguring it. We have clarified our documentation and code base to minimize this as a potential risk.

Our fixes consist, above all, of clarifications in the specification and usage documentation.

Additional organizations contributing to as core members of the development team, recommend addressing documentation issues above in advance of CNCF promotion

Since incubation we have added major contributors from Google, Intel, Verizon and More.

Consider integrations with other CI/CD projects

We have been heavily involved in integrations with SLSA, FRSCA, Tekton, TUF, SPIFFE/SPIRE, Keylime, Gitlab, SPDX, and other groups, as has been mentioned above.

Other References

[1] https://github.com/cncf/toc/blob/main/process/due-diligence-guidelines.md

[2] https://github.com/cncf/toc/blob/main/process/project_proposals.md

[3] https://github.com/cncf/toc/blob/main/PRINCIPLES.md

[4] https://github.com/cncf/toc/blob/f01a4fab58ee26280f93a40bb3962610820887e2/sigs/security-charter.md

[5] https://github.com/cncf/toc/pull/956