All things in-toto!

Supply chain attestations, policies, and adoption stories, oh my!

Marcela Melara (Intel Labs) & Santiago Torres-Arias (Purdue University)

Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

Software Supply Chain Attacks

Software Supply Chain Basics

software supply chain, noun

“a collection of systems, devices, organizations and people which produce a final software product”

Source: https://docs.lib.purdue.edu/ecepubs/160/

Software Supply Chain Attacks

software supply chain attack, noun

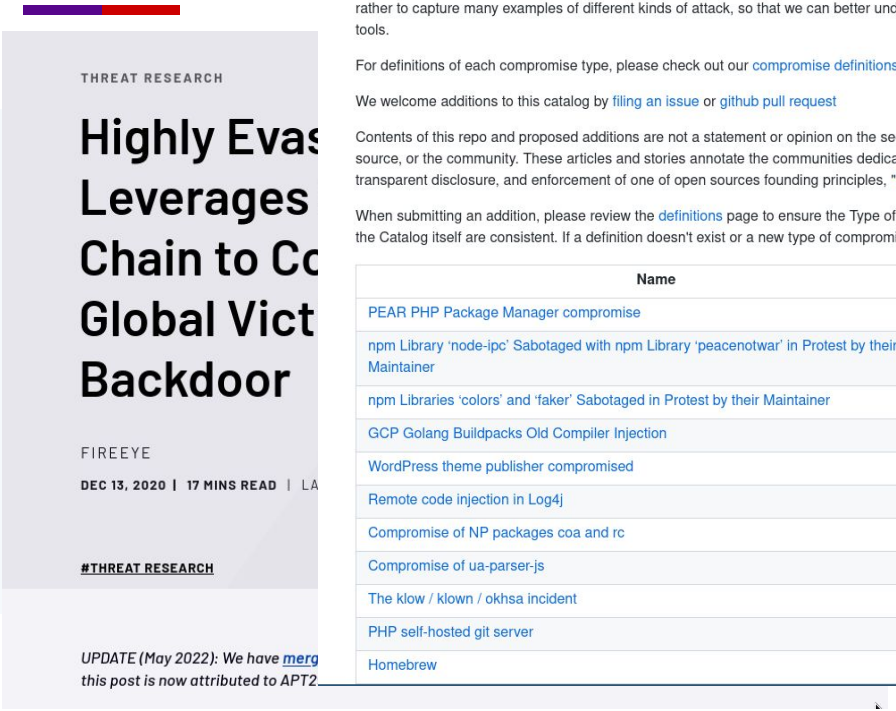
“when one or more weaknesses in the components of the software supply chain are compromised to introduce alterations into the final software product”

Source: https://docs.lib.purdue.edu/ecepubs/160/

742% □

Average annual increase in software supply chain attacks during 2019-2022

Source: Sonatype’s State of the Software Supply Chain 2022



Catalog of Supply Chain Compromises

This repository contains links to articles of software supply chain compromises. The goal is not to catalog every known supply chain attack, but rather to capture many examples of different kinds of attack, so that we can better understand the patterns and develop best practices and tools.

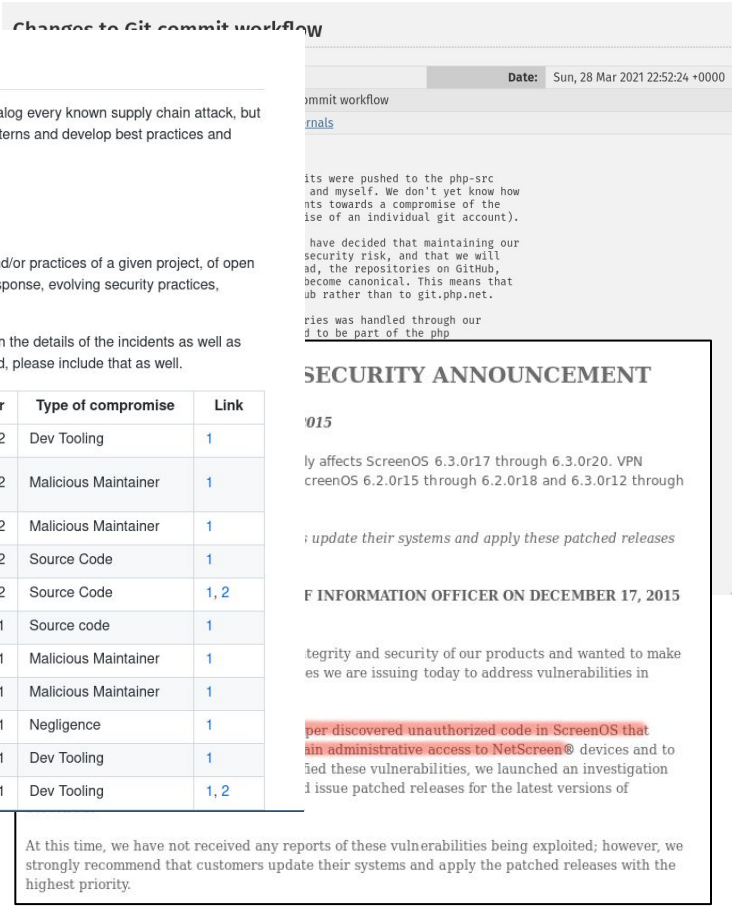
For definitions of each compromise type, please check out our compromise definitions page

We welcome additions to this catalog by filing an issue or github pull request

Contents of this repo and proposed additions are not a statement or opinion on the security stance and/or practices of a given project, of open source, or the community. These articles and stories annotate the communities dedication to rapid response, evolving security practices, transparent disclosure, and enforcement of one of open sources founding principles, "Linus's Law".

When submitting an addition, please review the definitions page to ensure the Type of Compromise on the details of the incidents as well as the Catalog itself are consistent. If a definition doesn't exist or a new type of compromise needs added, please include that as well.

Name	Year	Type of compromise	Link
PEAR PHP Package Manager compromise	2022	Dev Tooling	1
npm Library 'node-ipc' Sabotaged with npm Library 'peacenotwar' in Protest by their Maintainer	2022	Malicious Maintainer	1
npm Libraries 'colors' and 'faker' Sabotaged in Protest by their Maintainer	2022	Malicious Maintainer	1
GCP Golang Buildpacks Old Compiler Injection	2022	Source Code	1
WordPress theme publisher compromised	2022	Source Code	1, 2
Remote code injection in Log4j	2021	Source code	1
Compromise of NP packages coa and rc	2021	Malicious Maintainer	1
Compromise of ua-parser-js	2021	Malicious Maintainer	1
The klow / klown / okhsa incident	2021	Negligence	1
PHP self-hosted git server	2021	Dev Tooling	1
Homebrew	2021	Dev Tooling	1, 2

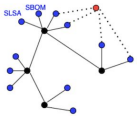


Ecosystem at a glance

Many Solutions

Evidence Gathering

Information Discovery



GUAC

SCITT

Policy / Validation

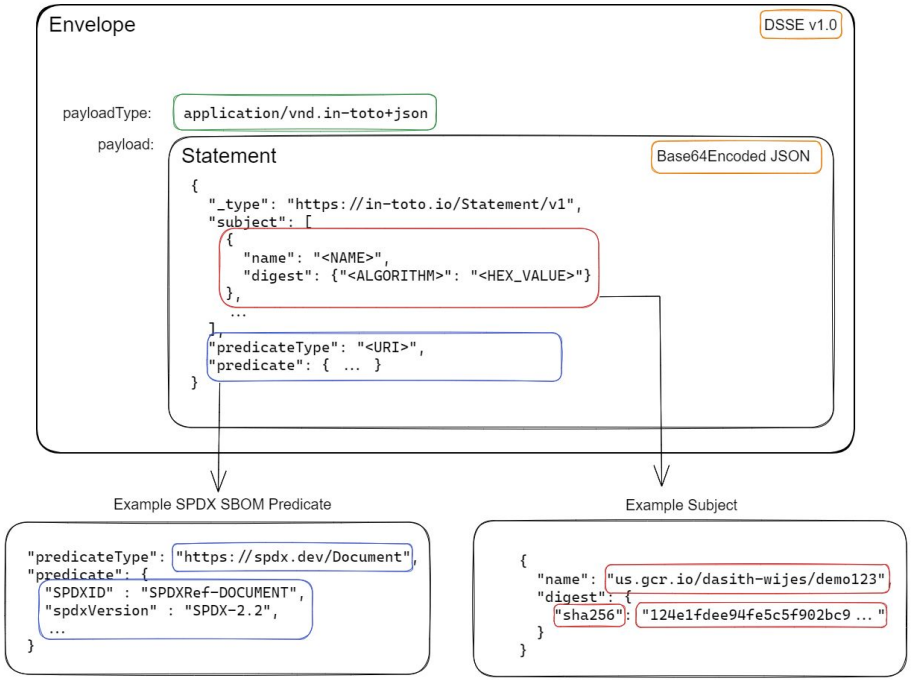
Open Policy Agent

Integrations & Adoptions

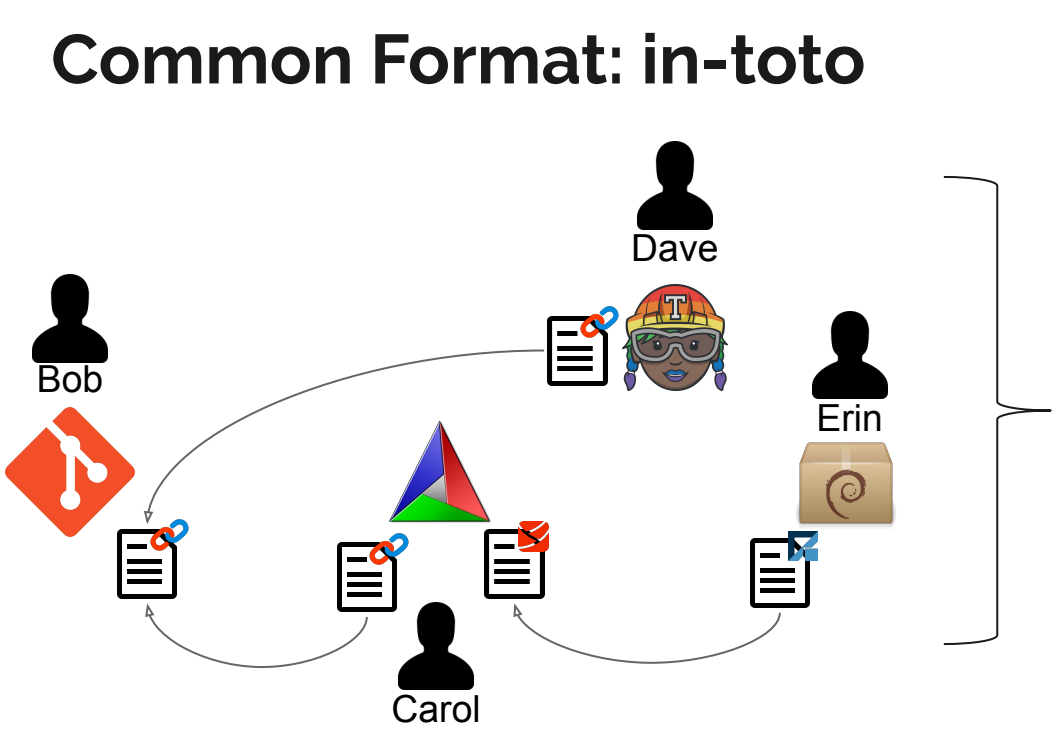
Chainloop
Toradex
Embedded. Computing.
solarwinds
Reproducible Builds
controlplane
TEKTON CHAINS
GUAC
SPDX
TUF
DATADOG
debian
CycloneDX
sigstore
spiffe
Jenkins

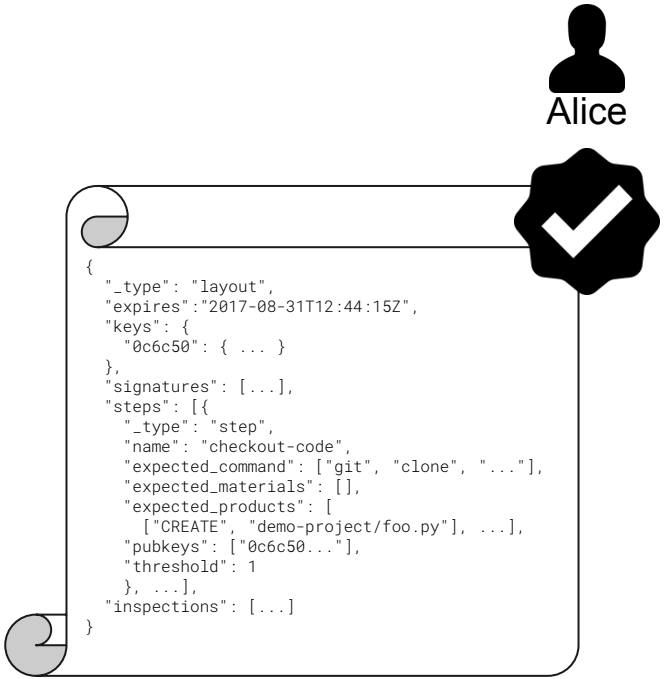
in-toto in a nutshell

Common Format: in-toto

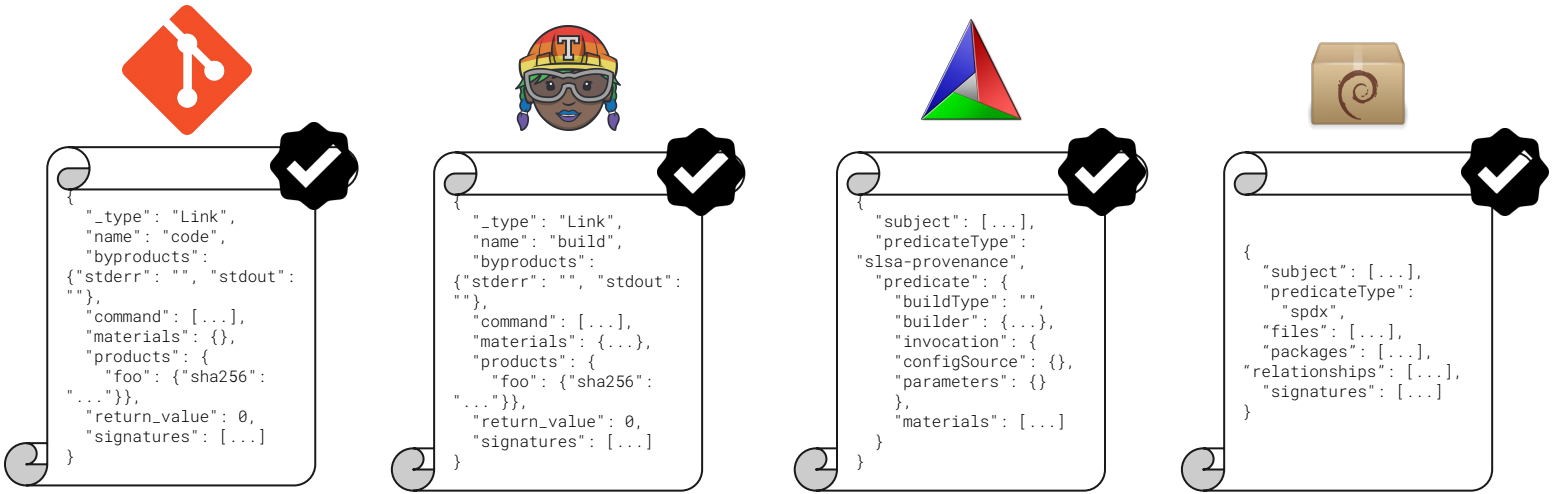


Common Format: in-toto



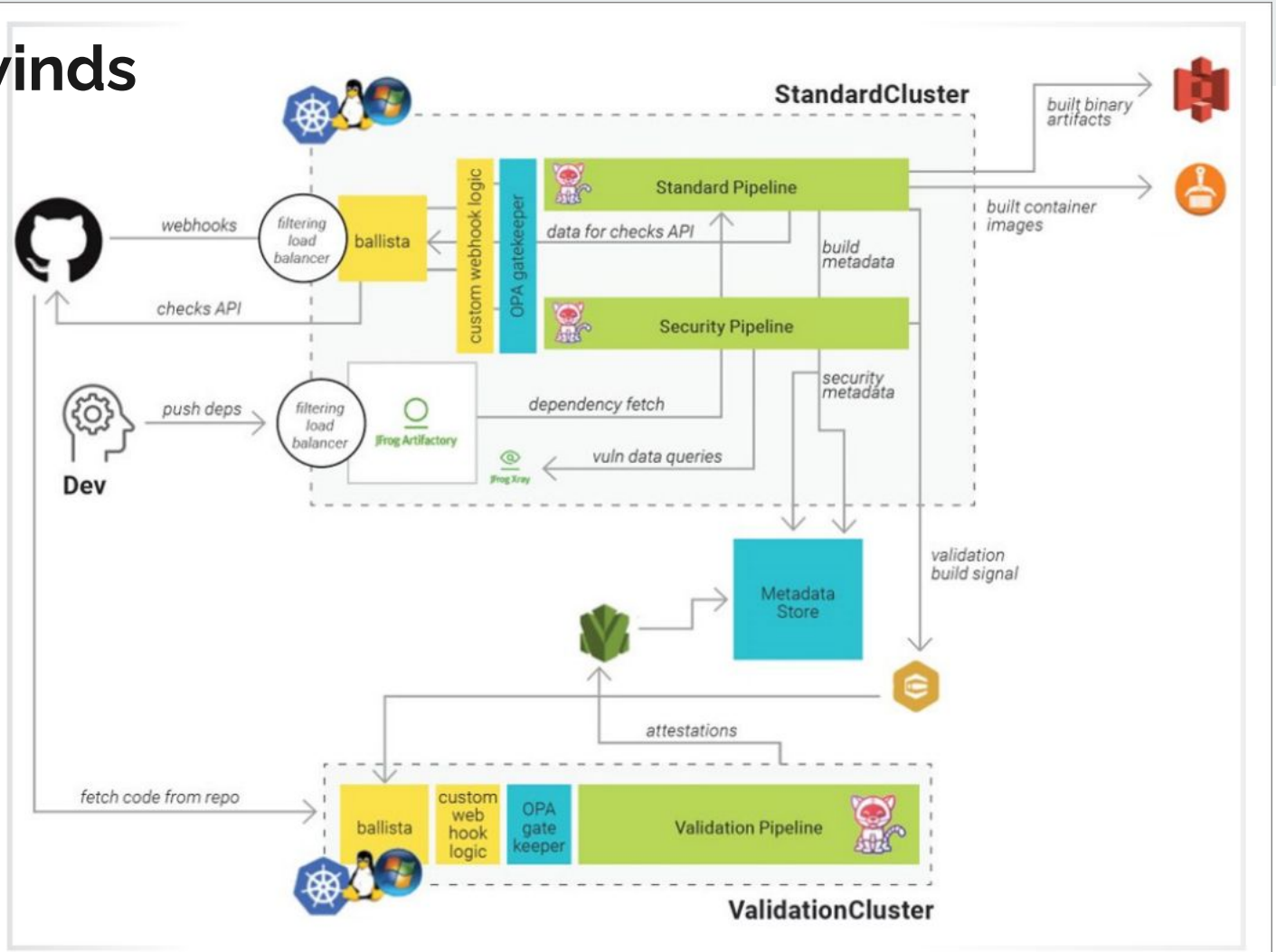


Common Format: in-toto

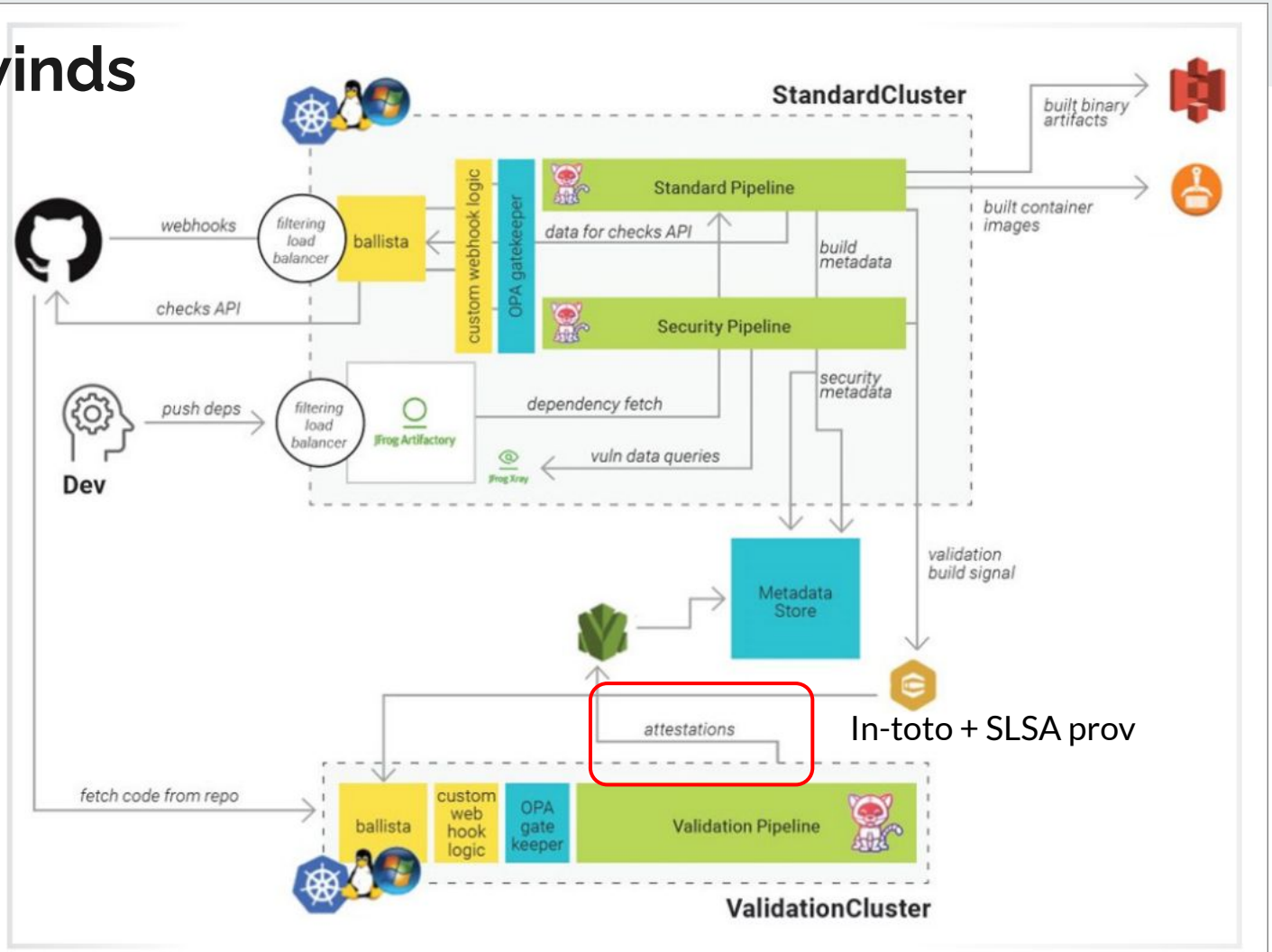


Security questions you can answer with in-toto: Predicates

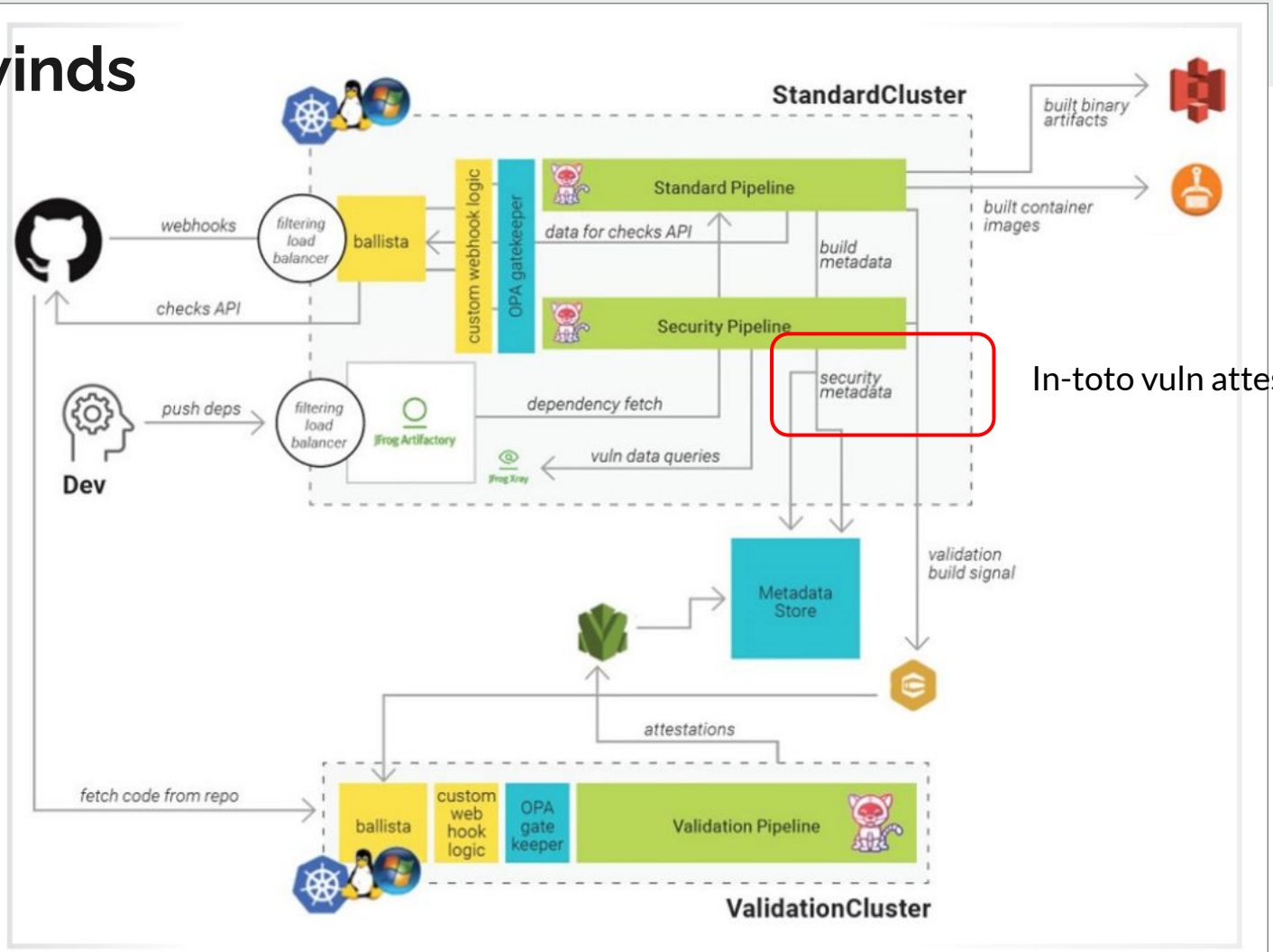
Solarwinds


StandardCluster
built binary artifacts
built container images
webhooks
filtering load balancer
ballista
custom webhook logic
OPA gatekeeper
Standard Pipeline
data for checks API
build metadata
checks API
Security Pipeline
security metadata
push deps
filtering load balancer
JFrog Artifactory
dependency fetch
vuln data queries
JFrog Xray
Dev
Metadata Store
validation build signal
attestations
fetch code from repo
ballista
custom web hook logic
OPA gate keeper
Validation Pipeline
ValidationCluster

Solarwinds

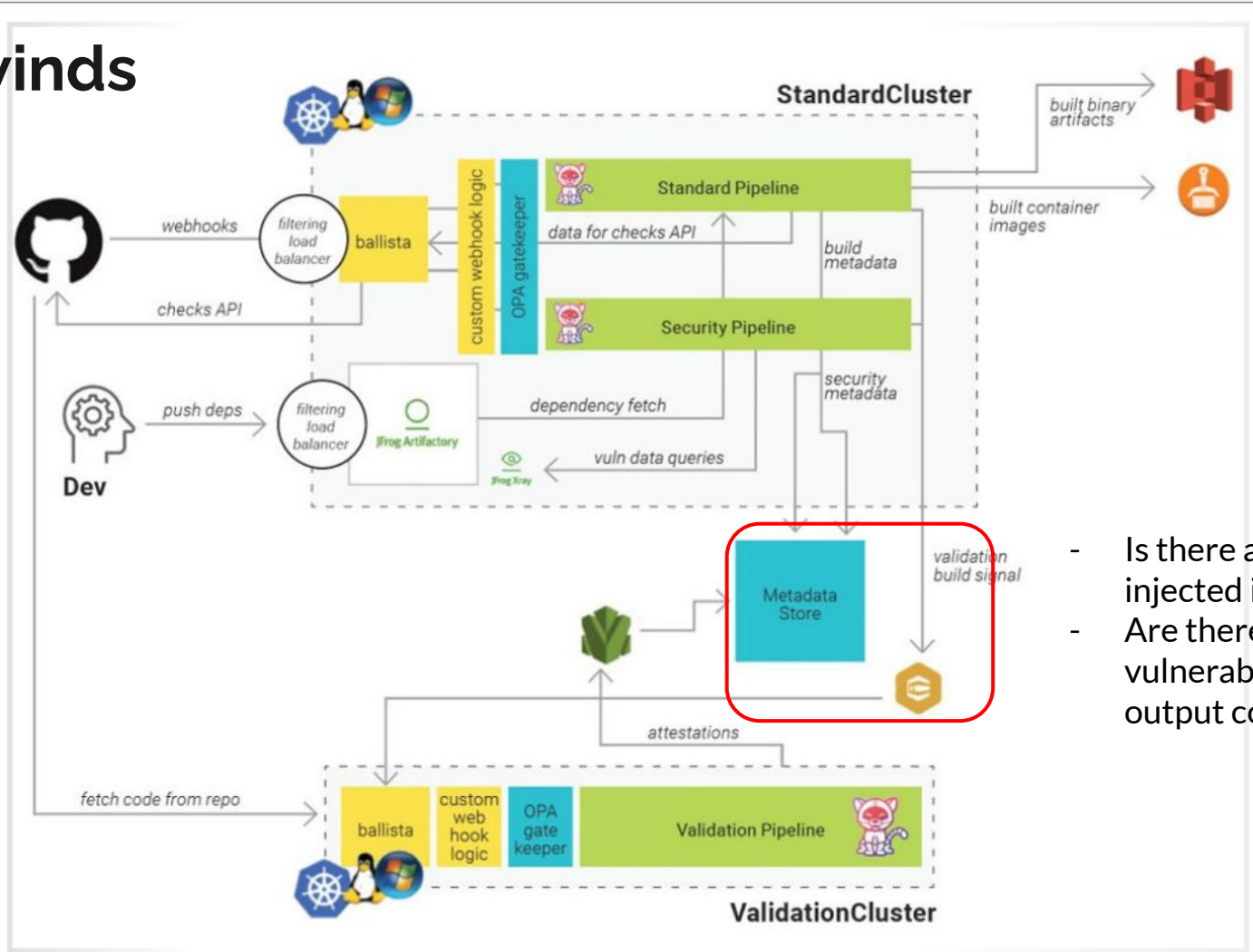

StandardCluster
built binary artifacts
built container images
webhooks
filtering load balancer
ballista
custom webhook logic
OPA gatekeeper
Standard Pipeline
data for checks API
build metadata
checks API
Security Pipeline
security metadata
push deps
filtering load balancer
JFrog Artifactory
dependency fetch
vuln data queries
JFrog Xray
Dev
Metadata Store
validation build signal
attestations
In-toto + SLSA prov
fetch code from repo
ballista
custom web hook logic
OPA gate keeper
Validation Pipeline
ValidationCluster

Solarwinds



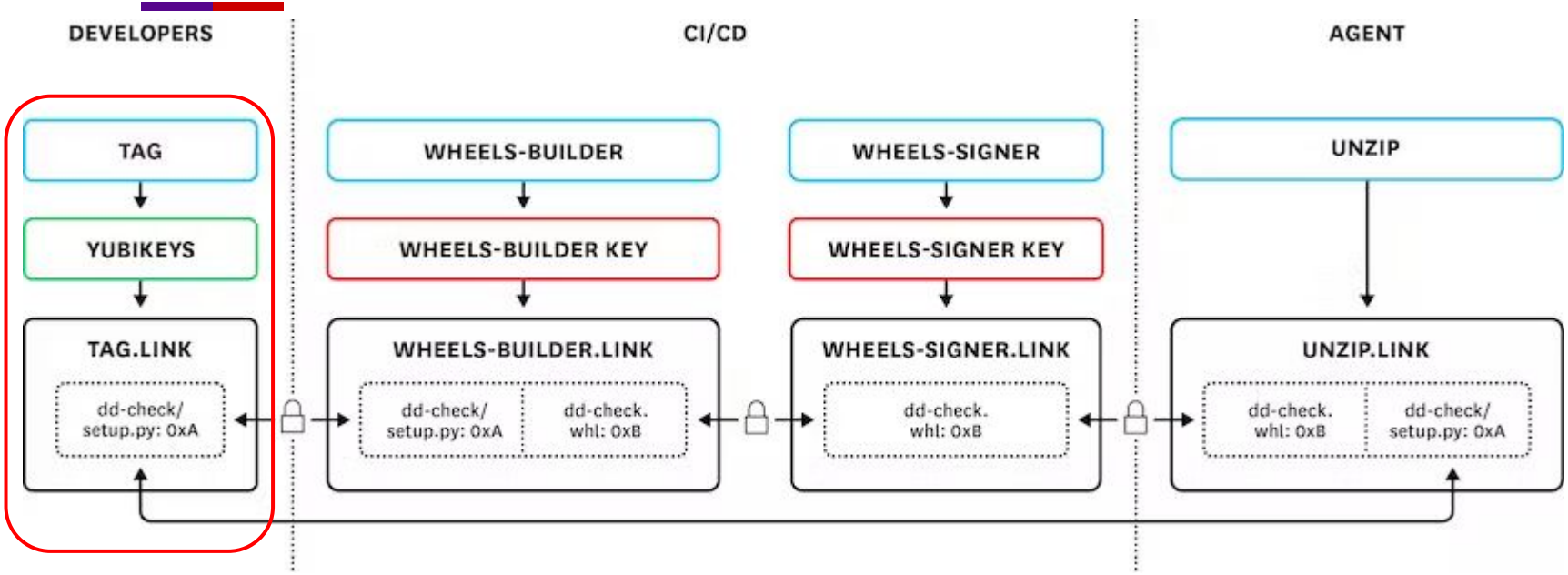
In-toto vuln attest?

Solarwinds



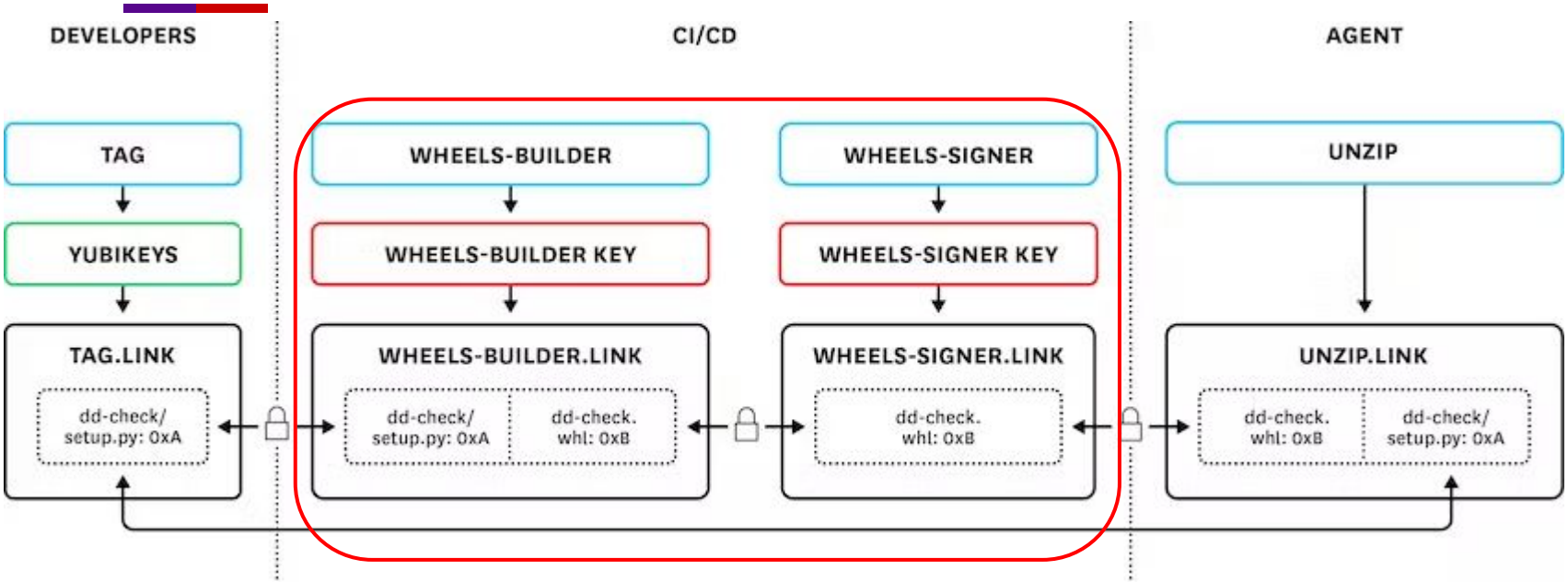
- Is there a backdoor injected in the build?
- Are there any vulnerabilities in the output code?

Datadog's Pipeline



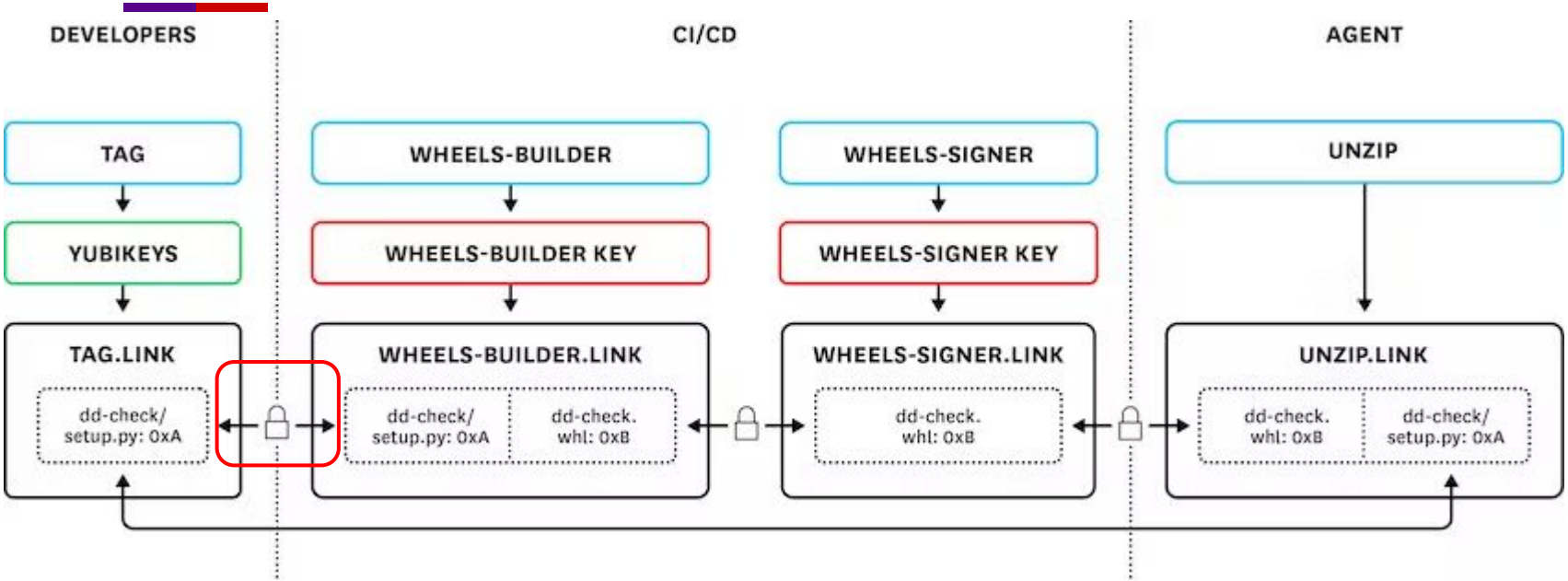
- Is the code released written by a known developer (using a hardware token?)

Datadog's Pipeline



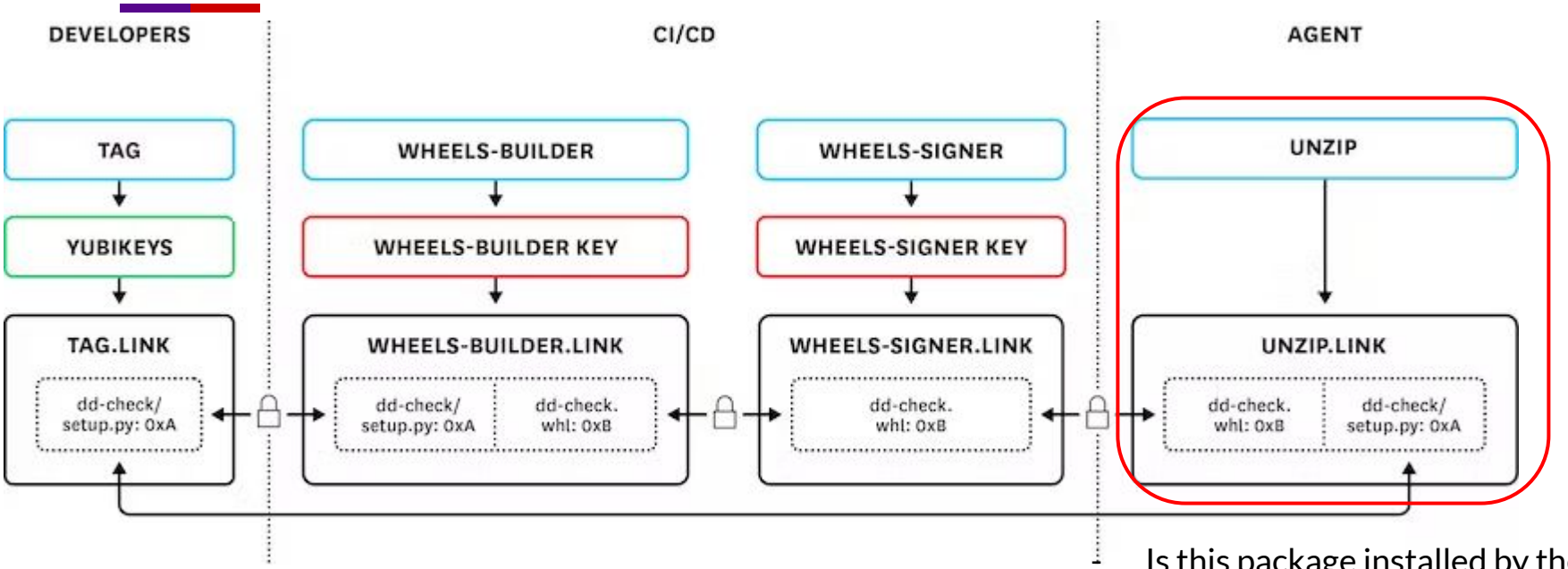
- Is the package built in a Datadog-controlled GitLab instance

Datadog's Pipeline



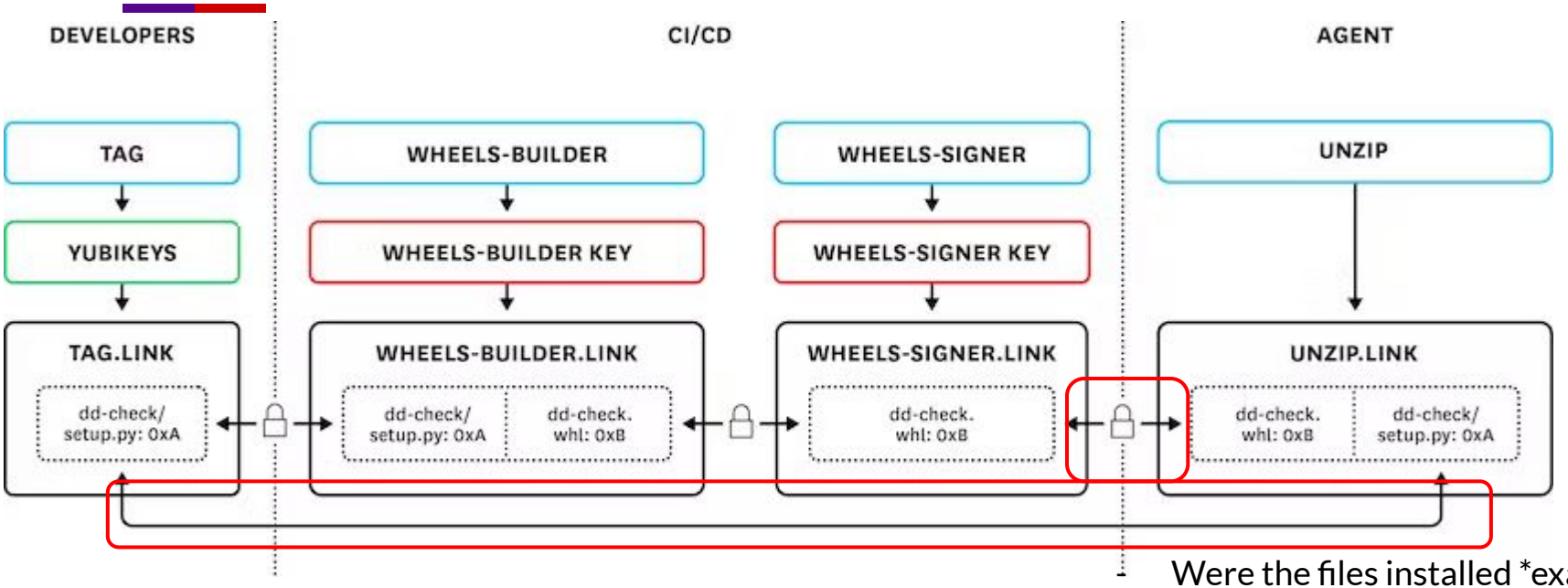
- Is the code built the one written by the developers?

Datadog's Pipeline



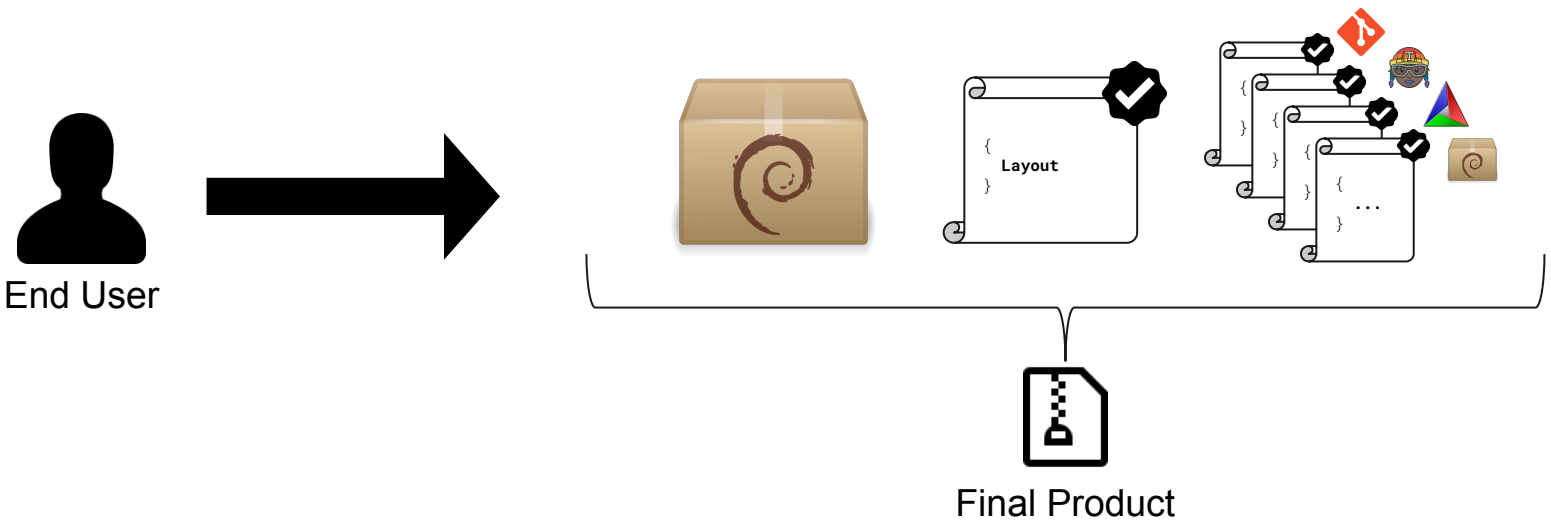
Is this package installed by the datadog agent?

Datadog's Pipeline



- Were the files installed *exactly* the files written by the original developer
- Or made by the build machine?

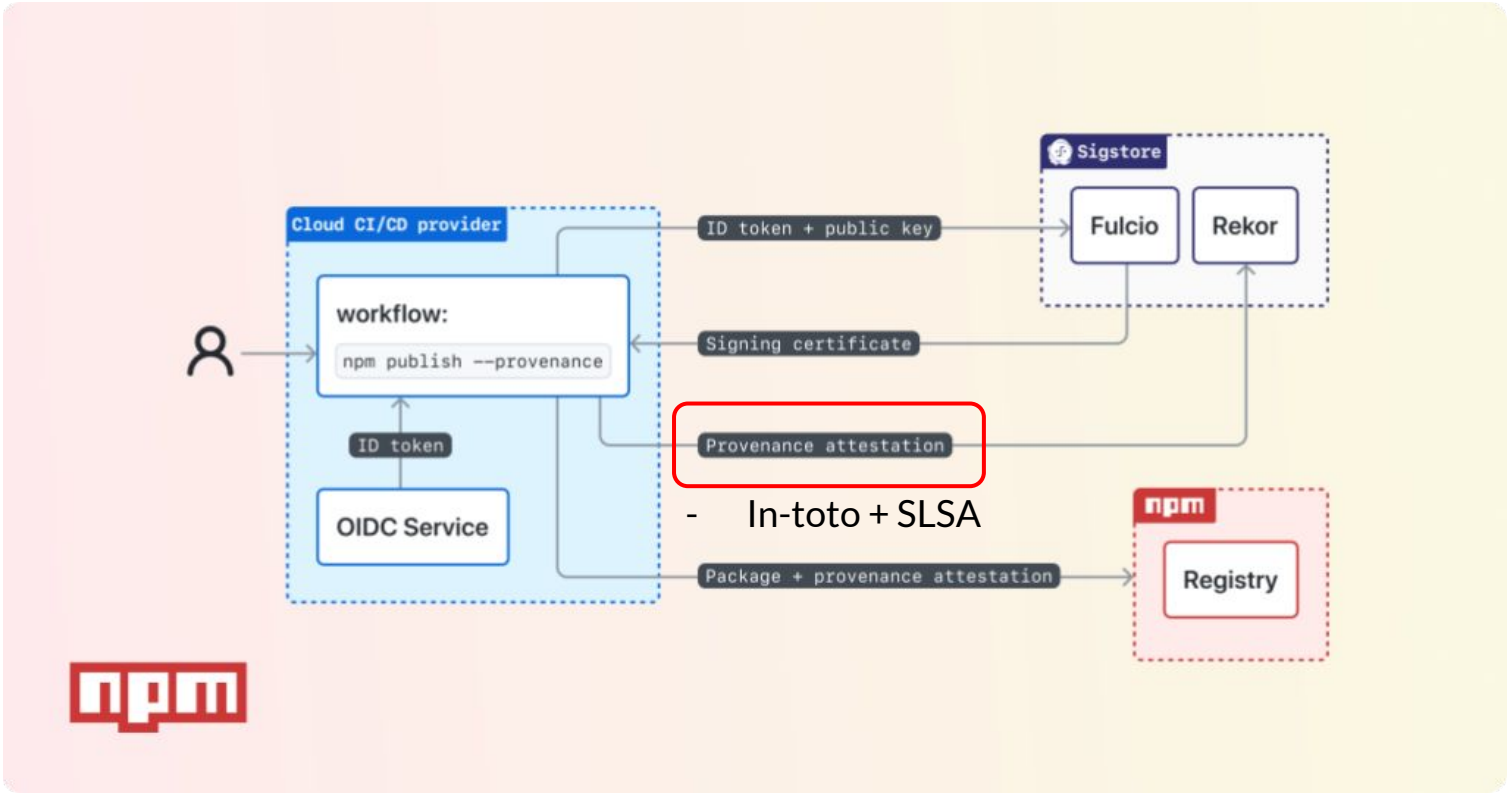
???


End User
Layout
Final Product

Github + NPM

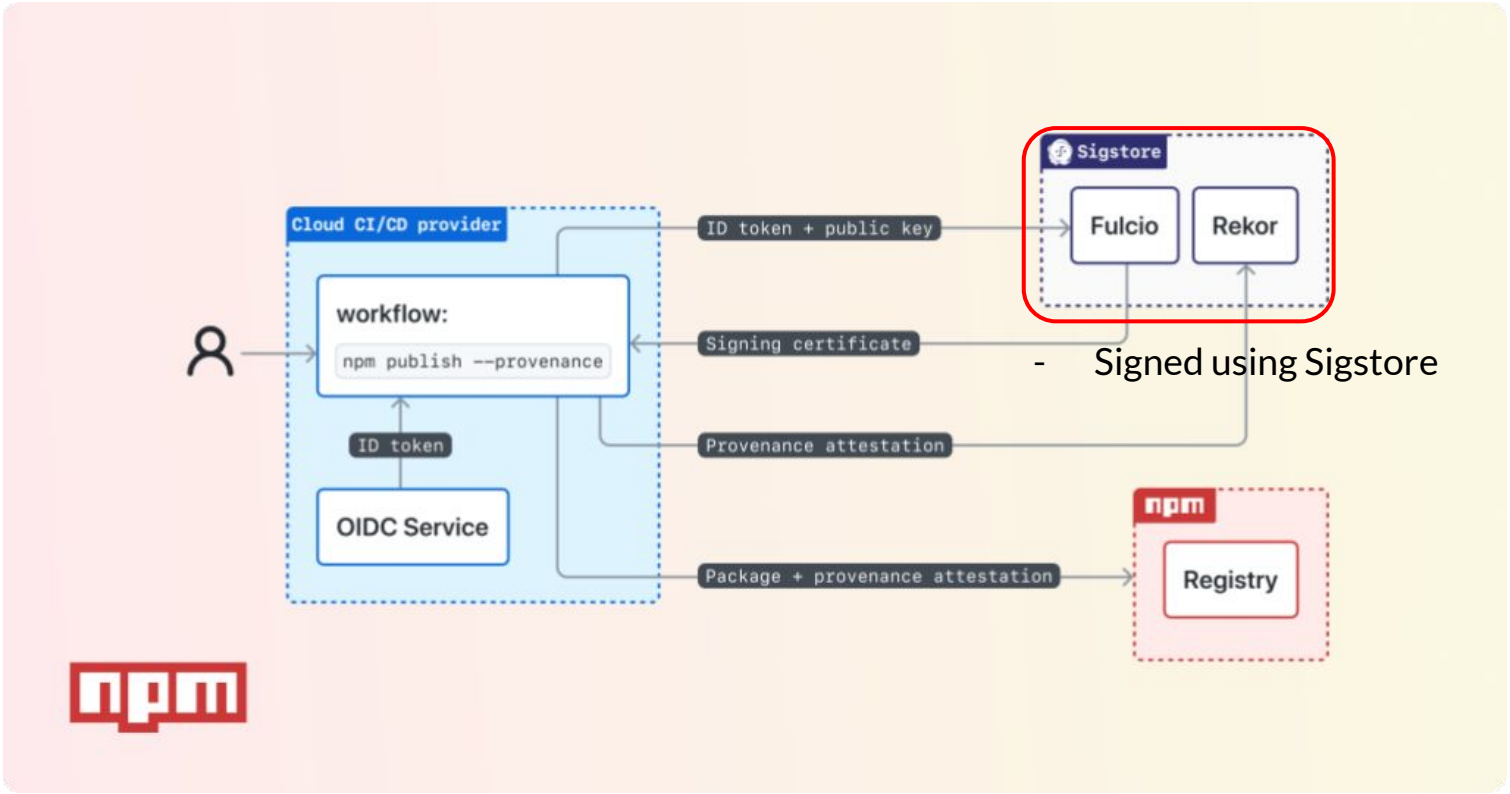
Provenance Beta

Github + NPM



- In-toto + SLSA

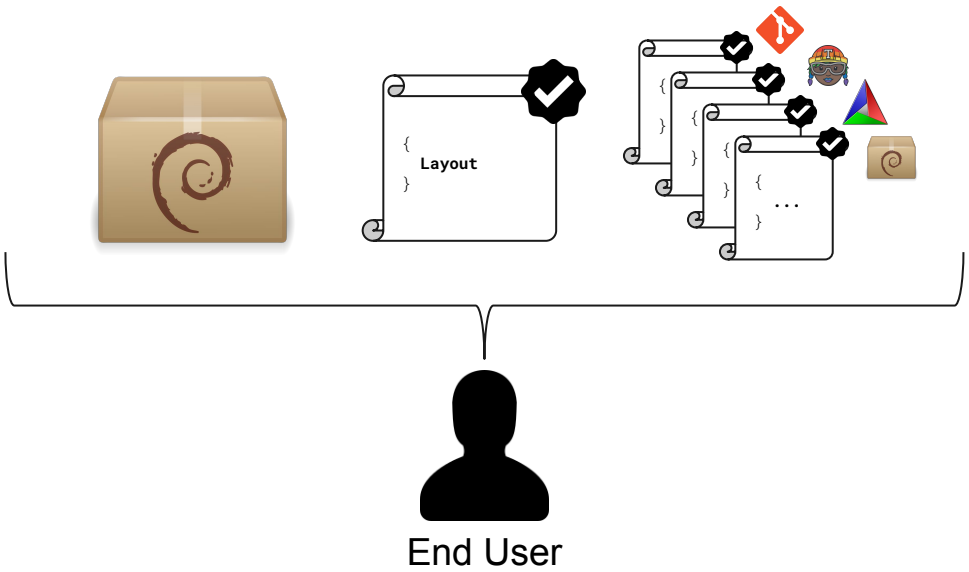
Github + NPM



- Signed using Sigstore

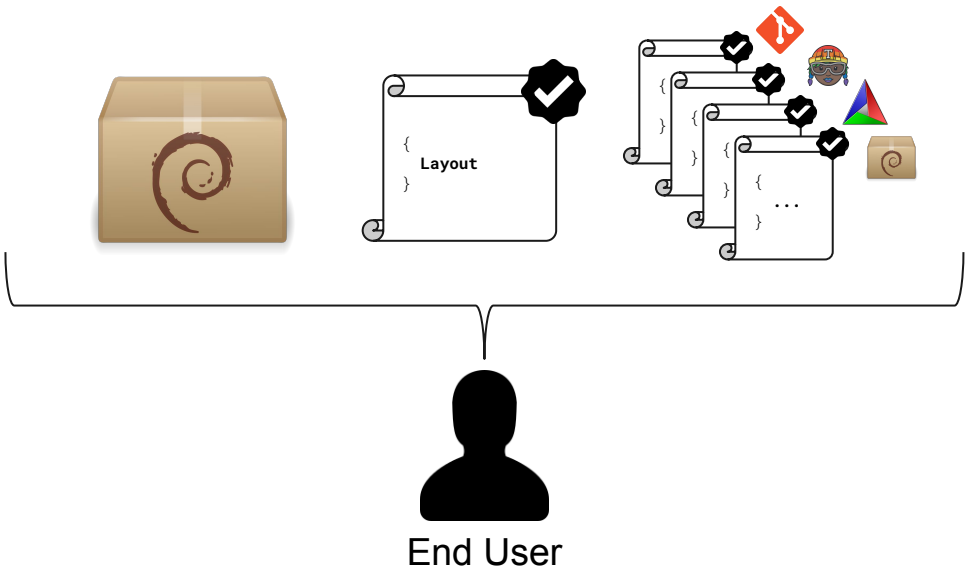
In-toto is the API of SW Supply Chain Security

- Who did what?
- To who?
- What were they using
- Where?
- Is this what they should be doing?



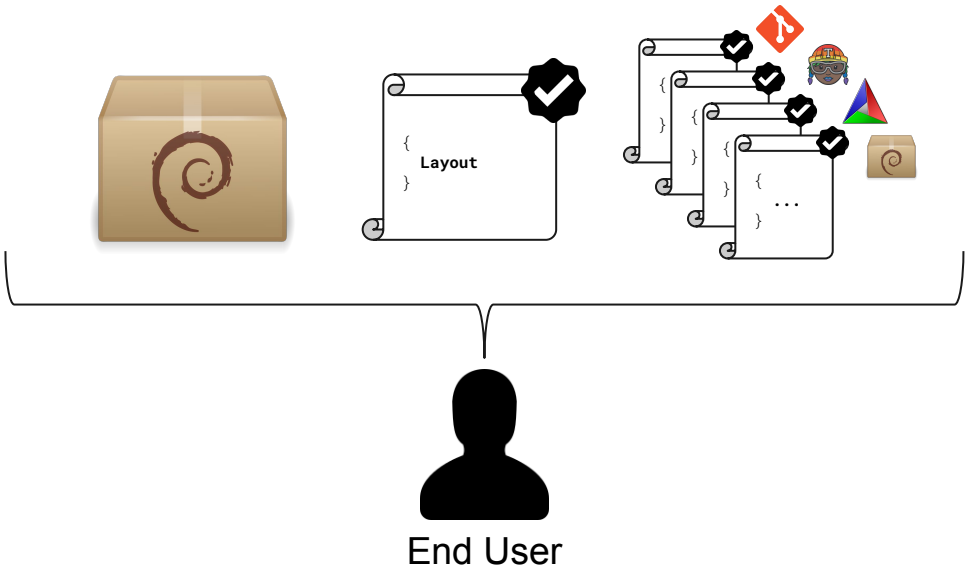
Things that you can ask (and answer)

- Did anybody run a vuln scan on this?



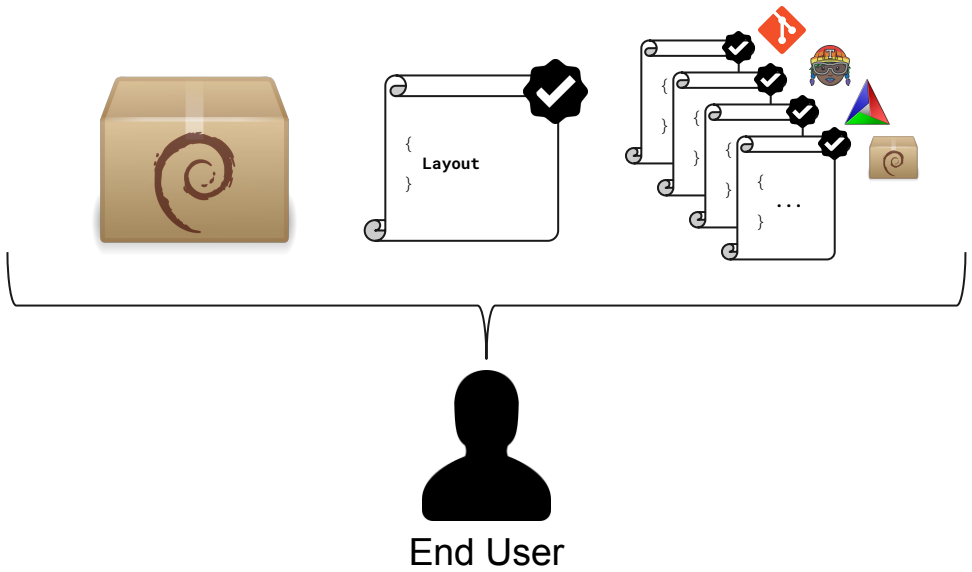
Things that you can ask (and answer)

- How was this built?



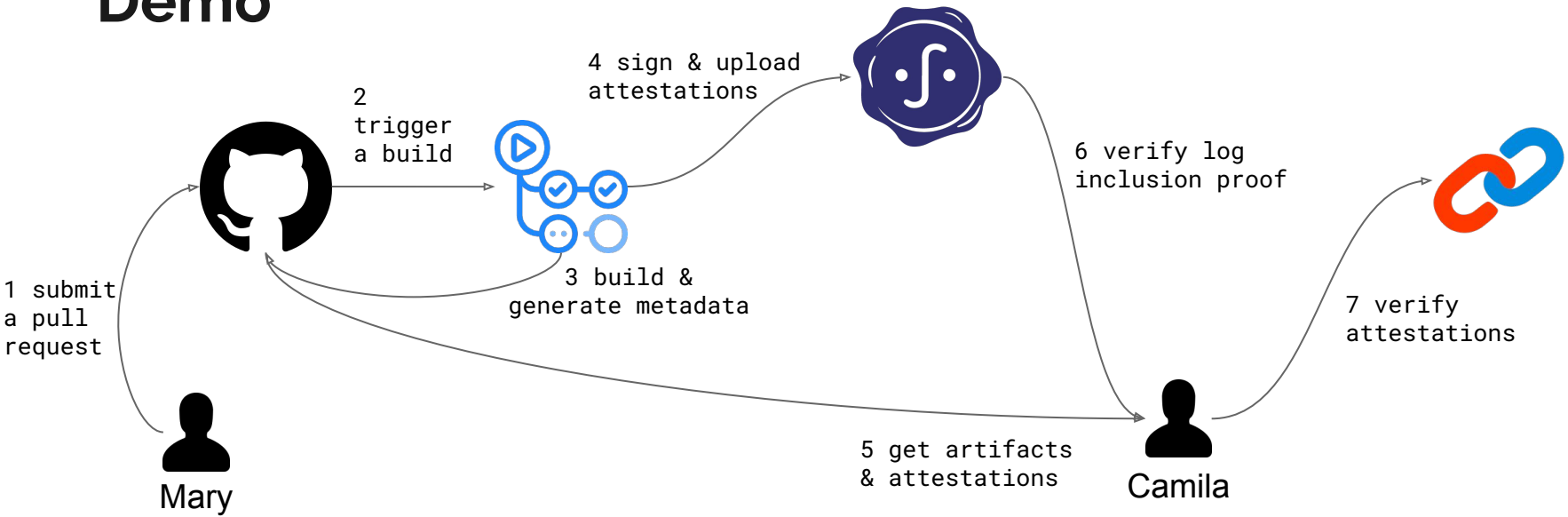
Things that you can ask (and answer)

- Do the tests pass?
- Was a runtime trace produced when building?
- Really anything!



Demo

Demo


1 submit
a pull
request
Mary
2
trigger
a build
3 build &
generate metadata
4 sign & upload
attestations
5 get artifacts
& attestations
6 verify log
inclusion proof
Camila
7 verify
attestations

Predicate spotlight: SCAI

Supply Chain Attribute Integrity:

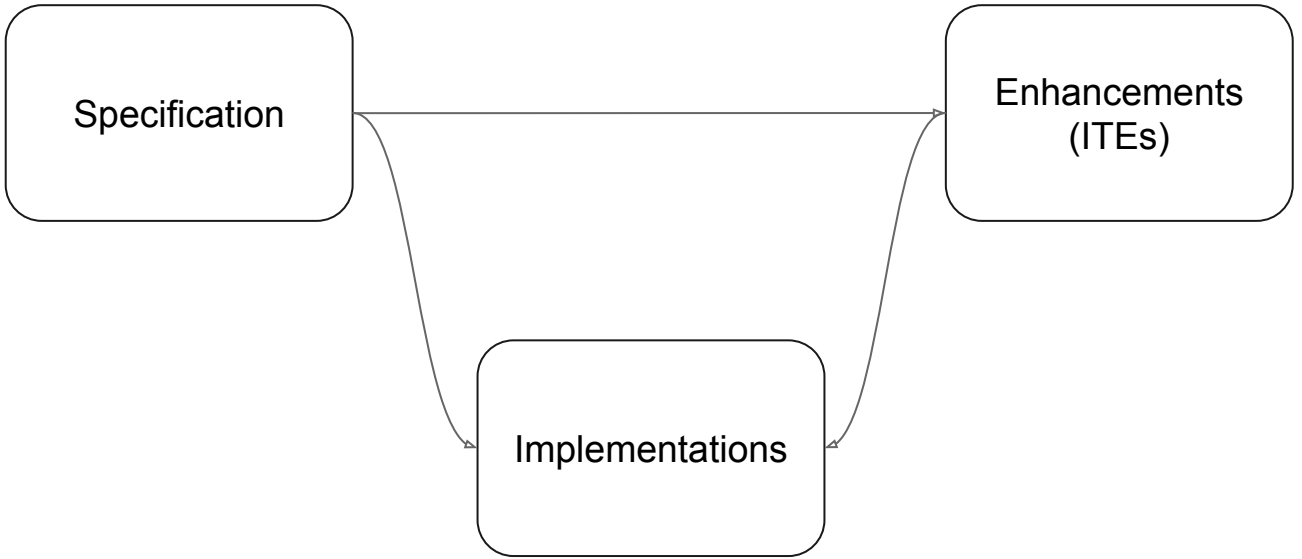
- Was the build tampered with?
- Was a legitimate version of the source repo used?
- Did the build produce an SBOM?
- Did the build produce SLSA Provenance?

```
{
    "predicateType": "https://in-toto.io/attestation/scai/attribute-report/v0.2",
    "predicate": {
        "attributes": [{
            "attribute": "<ATTRIBUTE>",
            "target": { [ResourceDescriptor] }, // optional
            "conditions": { /* object */ }, // optional
            "evidence": { [ResourceDescriptor] } // optional
        }],
        "producer": { [ResourceDescriptor] } // optional
    }
}
```

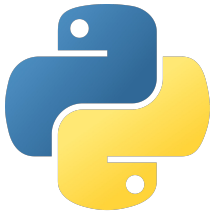
State of in-toto

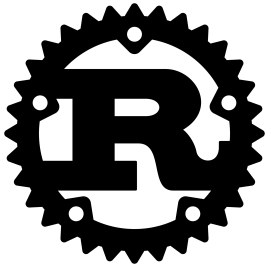
Incubation and Beyond

State of in-toto



State of in-toto





State of in-toto

in-toto Enhancements (ITEs)

Accepted

- ITE-1: in-toto Enhancement Format
- ITE-2: A general overview of combining TUF and in-toto to build compromise-resilient CI/CD
- ITE-3: Real-world example of combining TUF and in-toto for packaging Datadog Agent integrations
- ITE-4: Generic URI Schemes for in-toto
- ITE-5: Disassociate signature envelope specification from in-toto

Draft

- ITE-6: Generalized link format
- ITE-7: Signing & Verification With X509
- ITE-9: Introducing new in-toto Attestation types

ITE-6: Generalized Attestations

Attestation Spec

An in-toto **attestation** is authenticated metadata about one or more software artifacts, as per the SLSA Attestation Model. It has three layers that are independent but designed to work together:

- Envelope: Handles authentication and serialization.
- Statement: Binds the attestation to a particular subject and unambiguously identifies the types of the predicate.
- Predicate: Contains arbitrary metadata about the subject, with a type-specific schema.
- Bundle: Defines a method of grouping multiple attestations together.

The processing model provides pseudocode showing how these layers fit together.

See the top-level README for background and examples.

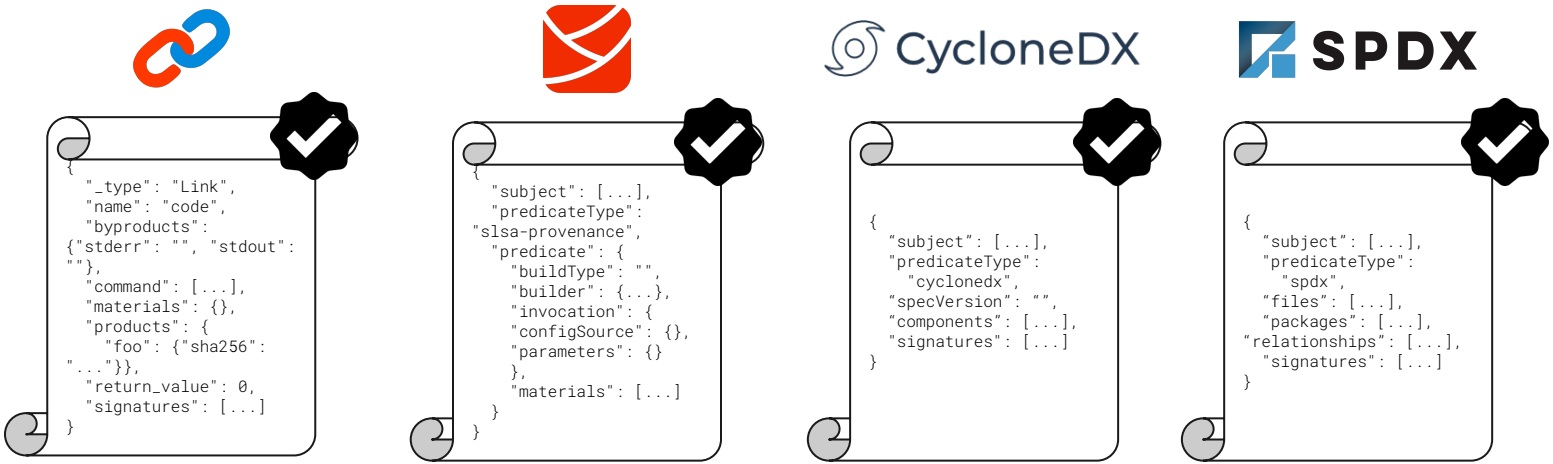
Parsing rules

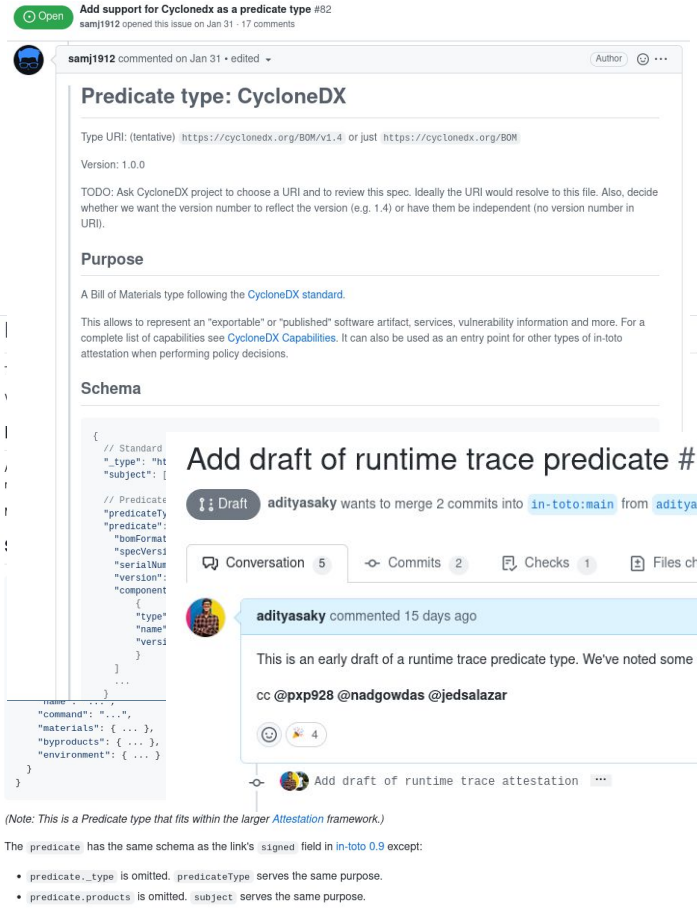
The following rules apply to Statement and predicates that opt-in to this model.

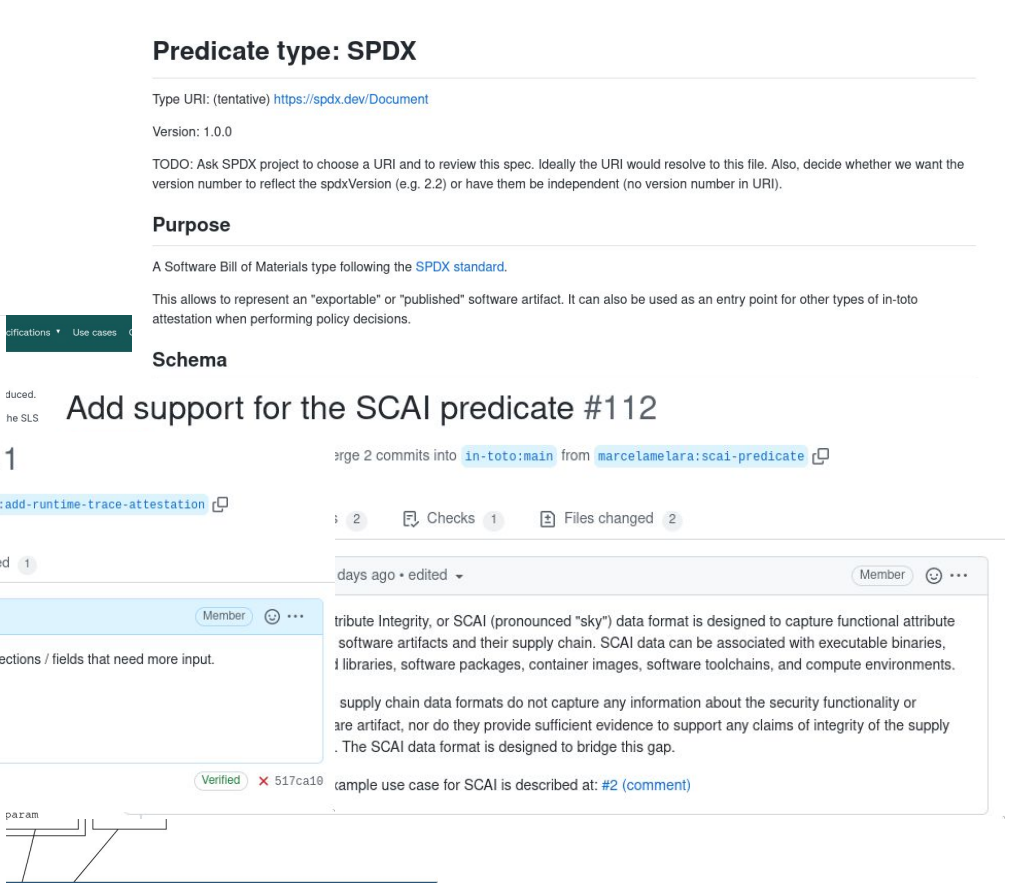
- **Unrecognized fields:** Consumers MUST ignore unrecognized fields. This is to allow minor version upgrades and extension fields. Ignoring fields is safe due to the monotonic principle.
- **Versioning:** Each type has a SemVer2 version number and the TypeURI reflects the major version number. A message is always semantically correct, but possibly incomplete, when parsed as any other version with the same major version number and thus the same TypeURI. Minor version changes always follow the monotonic principle. NOTE: 0.X versions are considered major versions.
- **Extension fields:** Producers MAY add extension fields to any JSON object by using a property name that is a TypeURI. The use of URI is to protect against name collisions. Consumers MAY parse and use these extensions if desired. The presence or absence of the extension field MUST NOT influence the meaning of any other field, and the field MUST follow the monotonic princple.
- **Monotonic:** A policy is considered monotonic if ignoring an attestation, or a field within an attestation, will never turn a DENY decision into an ALLOW. A predicate or field follows the monotonic principle if the expected policy that consumes it is monotonic. Consumers SHOULD design policies to be monotonic. Example: instead of "deny if a 'has vulnerabilities' attestation exists", prefer "deny unless a 'no vulnerabilities' attestation exists".

See versioning rules for details and examples.

ITE-6: in-toto as the Common Language




Open
Add support for Cyclonedx as a predicate type #82
samj1912 opened this issue on Jan 31 · 17 comments
samj1912 commented on Jan 31 • edited
Author
Predicate type: CycloneDX
Type URI: (tentative) https://cyclonedx.org/BOM/v1.4 or just https://cyclonedx.org/BOM
Version: 1.0.0
TODO: Ask CycloneDX project to choose a URI and to review this spec. Ideally the URI would resolve to this file. Also, decide whether we want the version number to reflect the version (e.g. 1.4) or have them be independent (no version number in URI).
Purpose
A Bill of Materials type following the CycloneDX standard.
This allows to represent an "exportable" or "published" software artifact, services, vulnerability information and more. For a complete list of capabilities see CycloneDX Capabilities. It can also be used as an entry point for other types of in-toto attestation when performing policy decisions.
Schema
(Note: This is a Predicate type that fits within the larger Attestation framework.)
The predicate has the same schema as the link's signed field in in-toto 0.9 except:
predicate._type is omitted. predicateType serves the same purpose.
predicate.products is omitted. subject serves the same purpose.


Predicate type: SPDX
Type URI: (tentative) https://spdx.dev/Document
Version: 1.0.0
TODO: Ask SPDX project to choose a URI and to review this spec. Ideally the URI would resolve to this file. Also, decide whether we want the version number to reflect the spdxVersion (e.g. 2.2) or have them be independent (no version number in URI).
Purpose
A Software Bill of Materials type following the SPDX standard.
This allows to represent an "exportable" or "published" software artifact. It can also be used as an entry point for other types of in-toto attestation when performing policy decisions.
Schema
Add draft of runtime trace predicate #111
Draft
adityasaky wants to merge 2 commits into in-toto:main from adityasaky:add-runtime-trace-attestation
Conversation 5
Commits 2
Checks 1
Files changed 1
adityasaky commented 15 days ago
Member
This is an early draft of a runtime trace predicate type. We've noted some subsections / fields that need more input.
cc @pxp928 @nadgowdas @jedsalazar
Add draft of runtime trace attestation
Verified
517ca10
Add support for the SCAI predicate #112
in-toto:main
marcelamelara:scai-predicate
Checks 1
Files changed 2
Member
tribute Integrity, or SCAI (pronounced "sky") data format is designed to capture functional attribute
software artifacts and their supply chain. SCAI data can be associated with executable binaries,
libraries, software packages, container images, software toolchains, and compute environments.
supply chain data formats do not capture any information about the security functionality or
are artifact, nor do they provide sufficient evidence to support any claims of integrity of the supply
. The SCAI data format is designed to bridge this gap.
ample use case for SCAI is described at: #2 (comment)

Defining New Attestation Formats

Process: ITE-9 – Introducing new in-toto Attestation types

Maintainers: Joshua Lock (Verizon), Marcela Melara (Intel), Mikhail Swift (TestifySec), Parth Patel (Kusari), Tom Hennen (Google)

Wrapping up

Looking ahead: More expressive policies!

- Need to extract value from richer attestation predicates
- ITE-10: Supporting Contextual in-toto Attestations in Layouts (Draft)
- ITE-11: Support attribute constraints in layouts (under review)

The in-toto Community

Invaluable Contributions

- Multiple ITEs

- Contributions to Implementations

- New in-toto Attestation Types

Join Us!

First Friday of every month

#in-toto @ CNCF Slack Workspace

#in-toto @ Libera.Chat IRC Server

in-toto-public@googlegroups.com

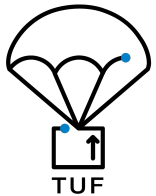
https://github.com/in-toto

in-toto and Mentoring

Join Us @ KubeCon + CloudNativeCon!

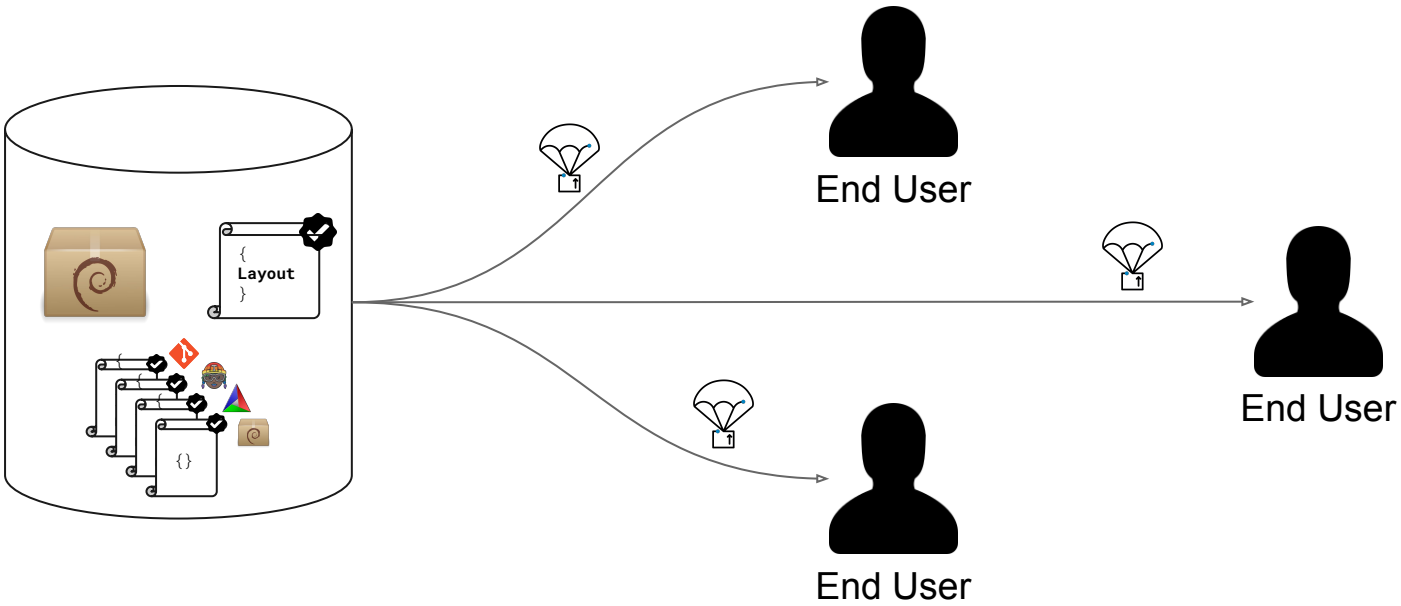
in-toto + TUF + Sigstore ContribFest

Thursday, October 27 at 2:30 PM - 4 PM EDT



Thank You! Questions?

ITE-2: TUF as a Root of Trust for in-toto



ITE-3: Datadog's Implementation of ITE-2

Secure Publication of Datadog Agent Integrations with TUF and in-toto

Trishank Karthik Kuppusamy

Published: June 3, 2019

ITE-4: Abstract Artifacts in in-toto

Sometimes Artifact != File

Generalized notions like Git objects, container images

Capture more abstract entities like merge requests

Or more granular – specific fields inside files

ITE-7: Support for X.509

ITE-7: Signing & Verification With X509

Table 1. Metadata

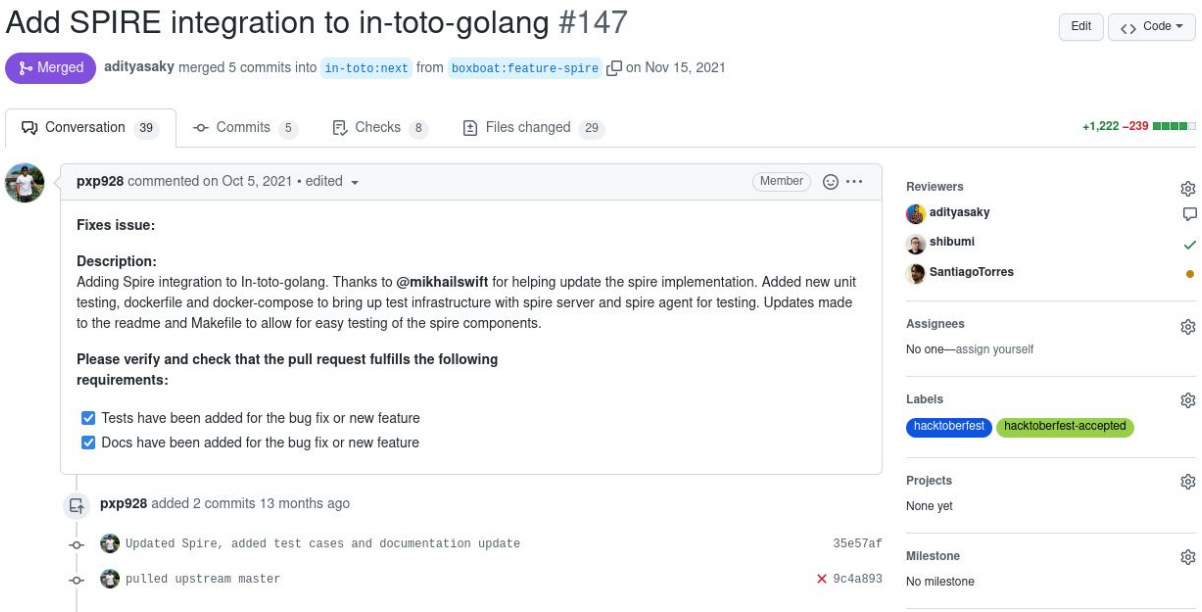
ITE	7
Title	Signing & Verification With X509
Sponsor	Mikhail Swift
Status	Draft
Type	Standards
Created	2021-07-01

Table of Contents

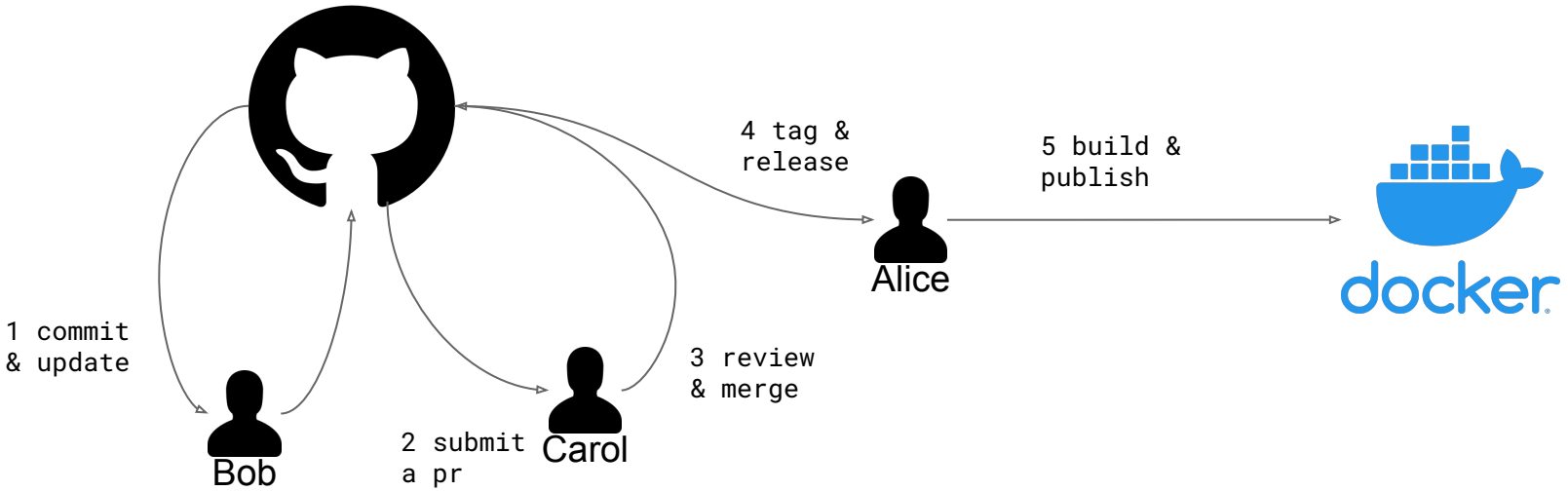
ITE-7: Support for X.509

Other PKI Setups

ITE-7: Support for X.509



Demo


1 commit
& update
Bob
2 submit
a pr
Carol
3 review
& merge
4 tag &
release
Alice
5 build &
publish
docker