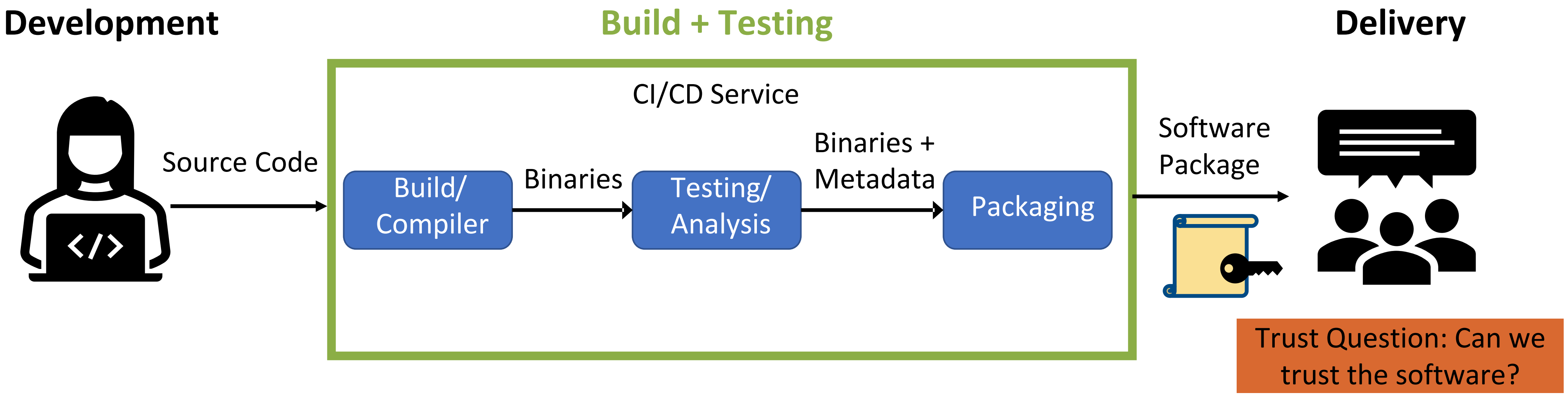
Intel Labs

Attribute-Based Integrity and Trust for the Software Supply Chain

Marcela Melara

Research Scientist, Trustworthy Distributed Systems, Security & Privacy Research Lab

SW Supply Chain Trust – Bird’s Eye View

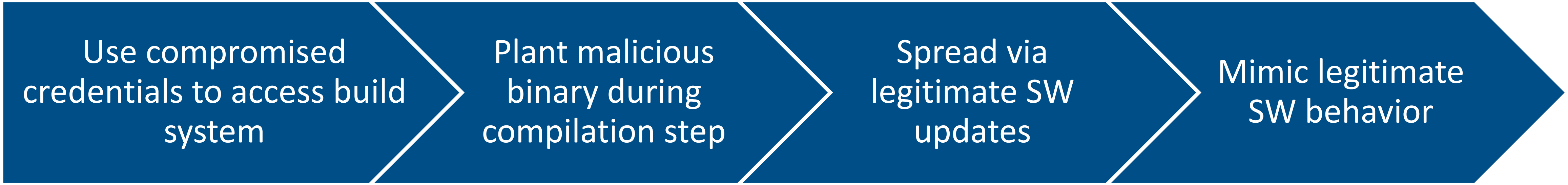


Trust Question: Can we trust the software?

Status Quo for SW Supply Chain Trust

- Establish identity-based trust in software artifacts
- For OSS: trust is institutional or reputational

A look at the SolarWinds Hack



Use compromised credentials to access build system

Plant malicious binary during compilation step

Spread via legitimate SW updates

Mimic legitimate SW behavior

Lessons from recent SW Supply Chain Attacks

- Code coming from reputable source is not always trustworthy
- Identity is insufficient, need details about code properties

What does it *really* mean to trust software?

- Actually asking:
  - Who wrote this software?
  - Who built this software?
  - What components make up this software?
  - How was the software built?
  - What platform was the software built on?
  - Was the build tampered with?
  - Was a legitimate version of gcc used?
  - Does the software contain buffer overflow vulnerabilities?
  - Does the software contain data race bugs?
  - etc

Code Signatures

SBOM + SLSA

Attribute-based Integrity

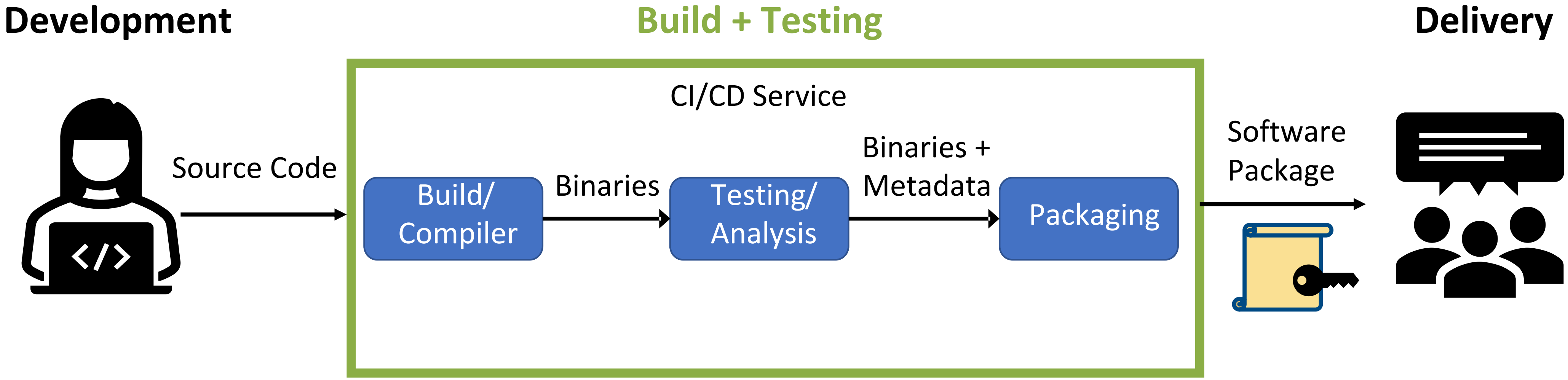
Key Insight:

We can collect an extra layer of information about code behavior *before it’s deployed*.

A Case for Attribute-based trust

- Good news: Info is already available through the supply chain!
- Examples:
  - Vulnerability analysis
  - Static code analysis
  - ML-based code analysis
  - Runtime traces of build systems
  - Compiler flags that affect code properties

Capturing fine-grained code attributes



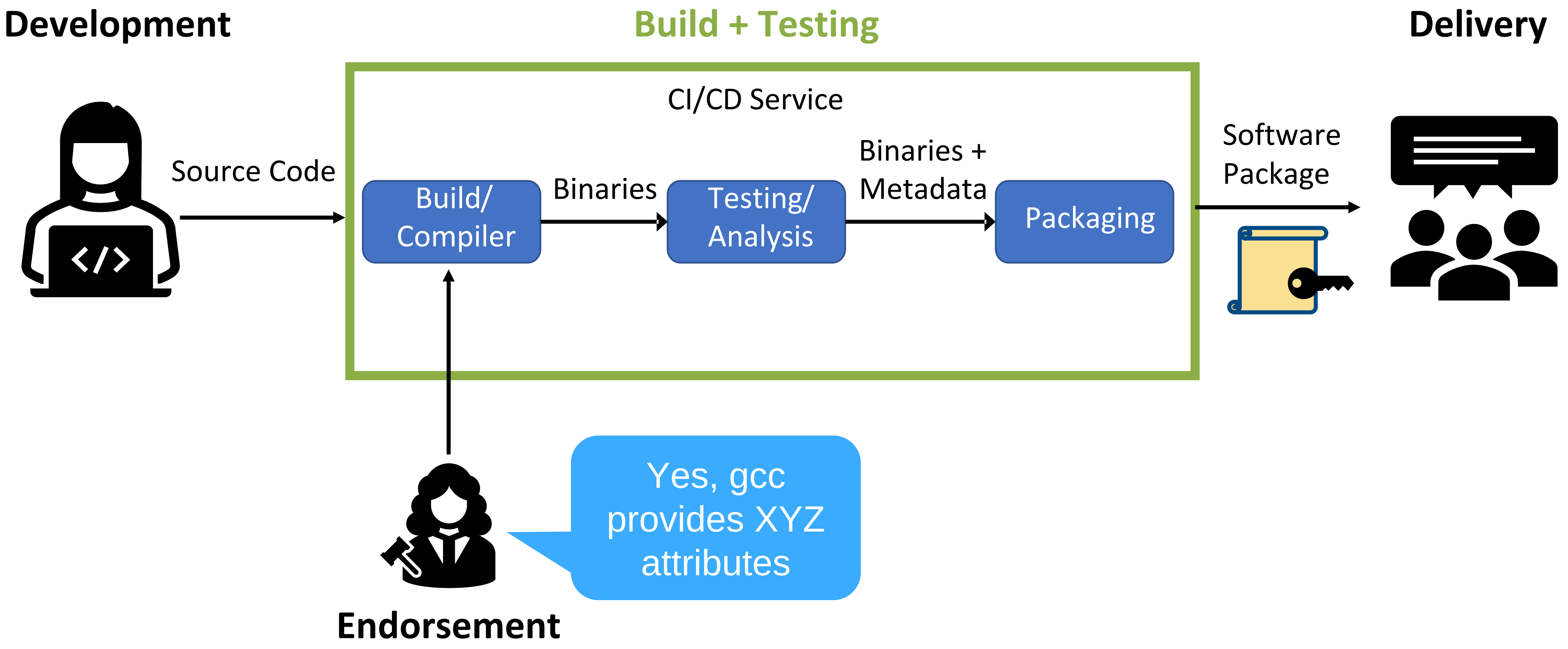
Attribute assertion:
If compiler configured with -fstack-protector flag,
Then binary has buffer overflow protection

SCAI: Supply Chain Attribute Integrity

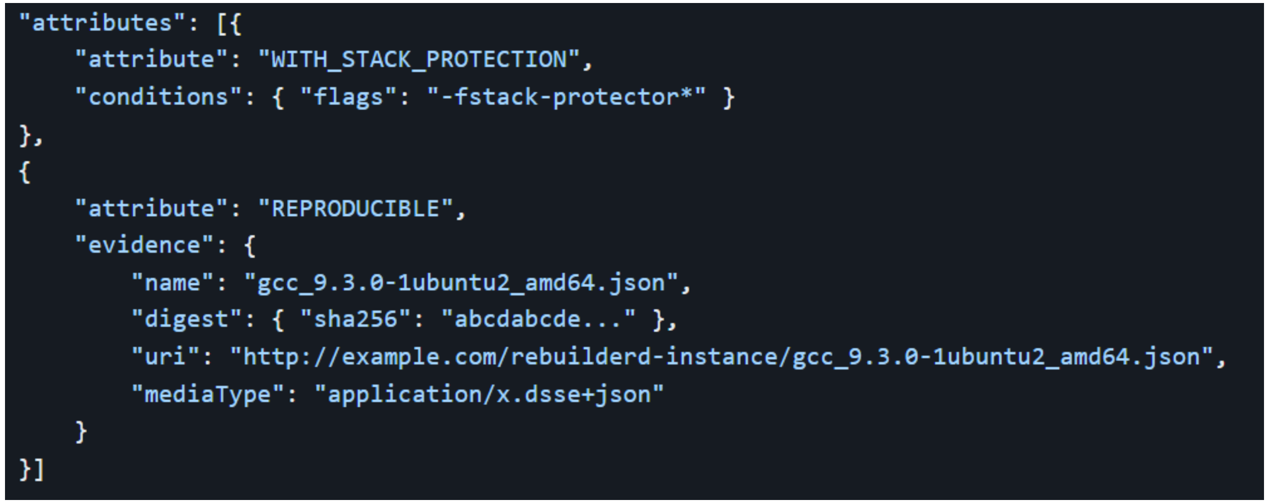
- Data format for asserting attributes and integrity information about software artifacts and the compute stack that produced them.
- Key Features:

1. General-Purpose: Single data format for any set of SW attributes
2. Evidence-based: Makes evidence for claims first-class primitive
3. Interoperable: Complements existing integrity frameworks

Capturing third-party endorsements



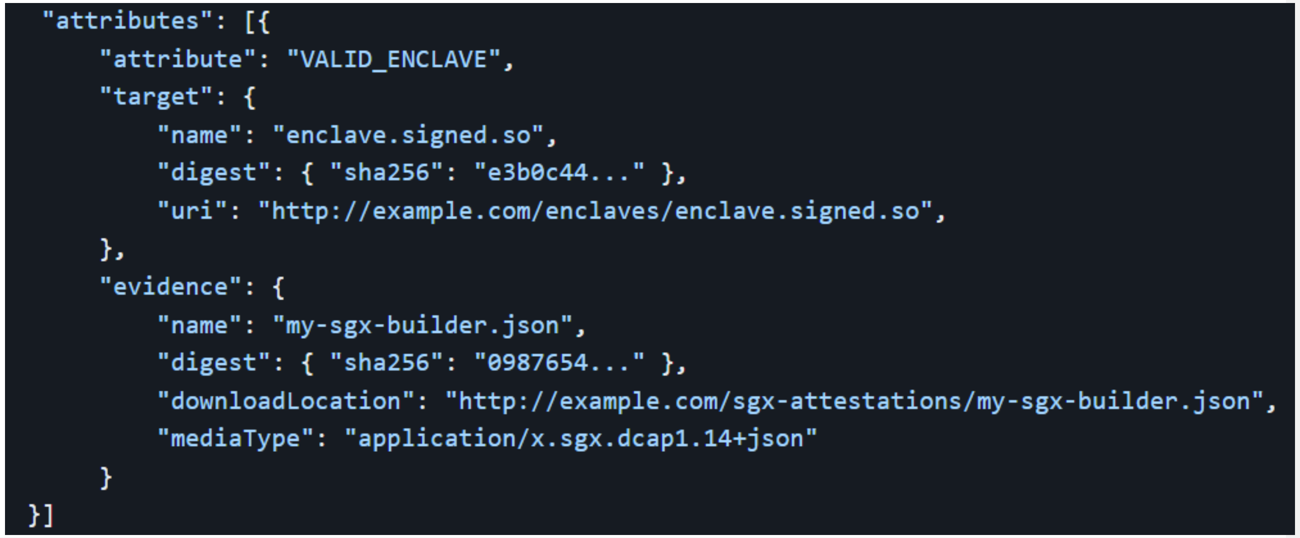
Example SCAI Attribute Assertions



```
"attributes": [{
    "attribute": "WITH_STACK_PROTECTION",
    "conditions": { "flags": "-fstack-protector*" }
},
{
    "attribute": "REPRODUCIBLE",
    "evidence": {
        "name": "gcc_9.3.0-1ubuntu2_amd64.json",
        "digest": { "sha256": "abcdabcde..." },
        "uri": "http://example.com/rebuilderd-instance/gcc_9.3.0-1ubuntu2_amd64.json",
        "mediaType": "application/x.dsse+json"
    }
}]
```

gcc compiler attributes

Valid hardware enclave



```
"attributes": [{
    "attribute": "VALID_ENCLAVE",
    "target": {
        "name": "enclave.signed.so",
        "digest": { "sha256": "e3b0c44..." },
        "uri": "http://example.com/enclaves/enclave.signed.so",
    },
    "evidence": {
        "name": "my-sgx-builder.json",
        "digest": { "sha256": "0987654..." },
        "downloadLocation": "http://example.com/sgx-attestations/my-sgx-builder.json",
        "mediaType": "application/x.sgx.dcap1.14+json"
    }
}]
```

BUT:

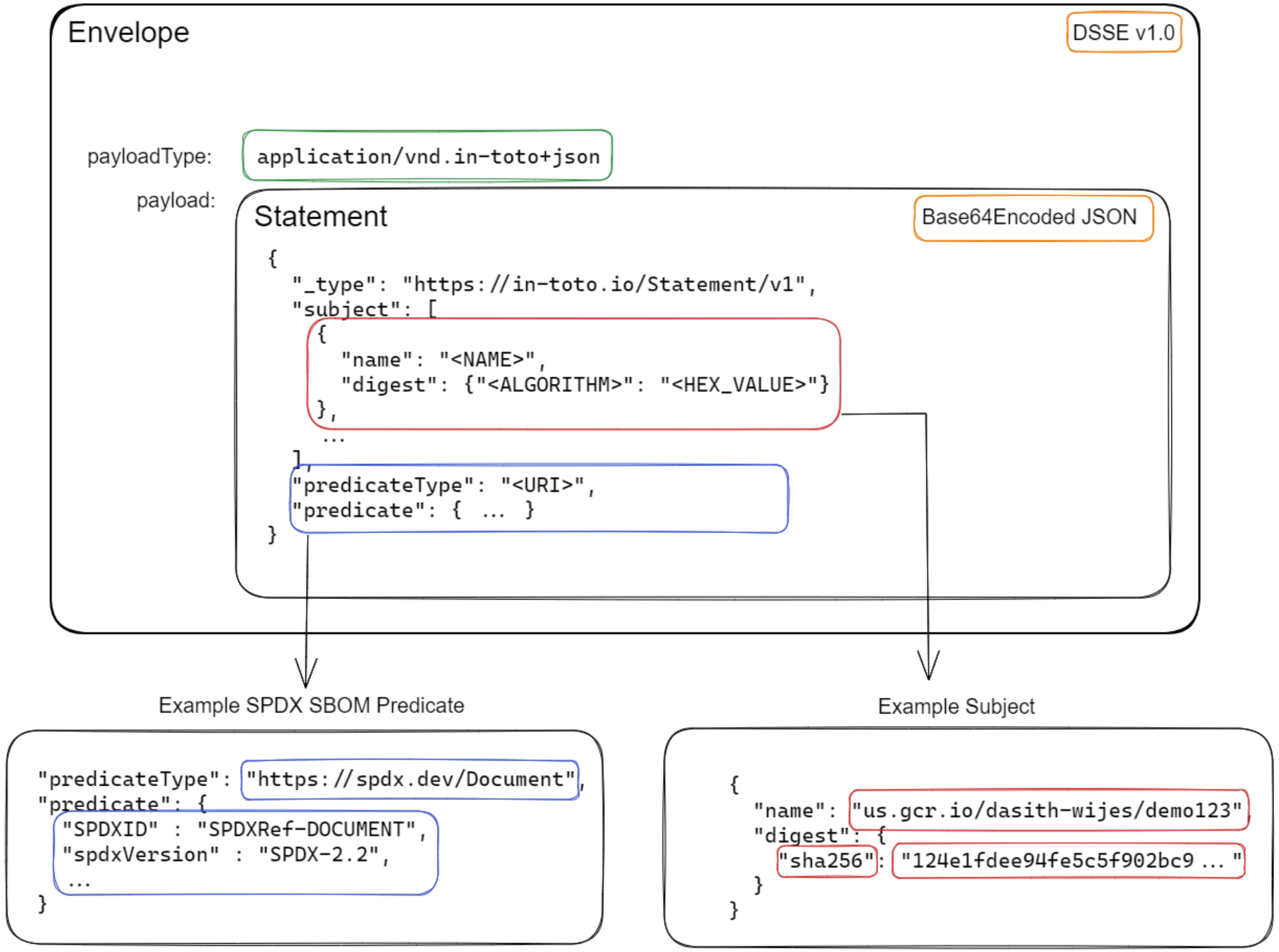
We also need to know the ***authenticity*** of the SWSC metadata.

in-toto Framework (CNCF project)

- Goal: Authenticated claims about the SW supply chain
- Two use cases: Regulatory compliance and incident response
- Integrated SCAI with in-toto
  - Standard data format for claims about any aspect of the SW supply chain
  - Production-ready: Adopters include GitHub, GitLab, npm, IBM

CNCF = Cloud Native Compute Foundation

in-toto Format



Envelope

DSSE v1.0

payloadType: `application/vnd.in-toto+json`

payload:

Statement

Base64Encoded JSON

```
{
  "_type": "https://in-toto.io/Statement/v1",
  "subject": [
    {
      "name": "<NAME>",
      "digest": {"<ALGORITHM>": "<HEX_VALUE>"}
    },
    ...
  ],
  "predicateType": "<URI>",
  "predicate": { ... }
}
```

Example SPDX SBOM Predicate

```
"predicateType": "https://spdx.dev/Document",
"predicate": {
  "SPDXID" : "SPDXRef-DOCUMENT",
  "spdxVersion" : "SPDX-2.2",
  ...
}
```

Example Subject

```
{
  "name": "us.gcr.io/dasith-wijes/demo123",
  "digest": {
    "sha256": "124e1fdee94fe5c5f902bc9 ... "
  }
}
```

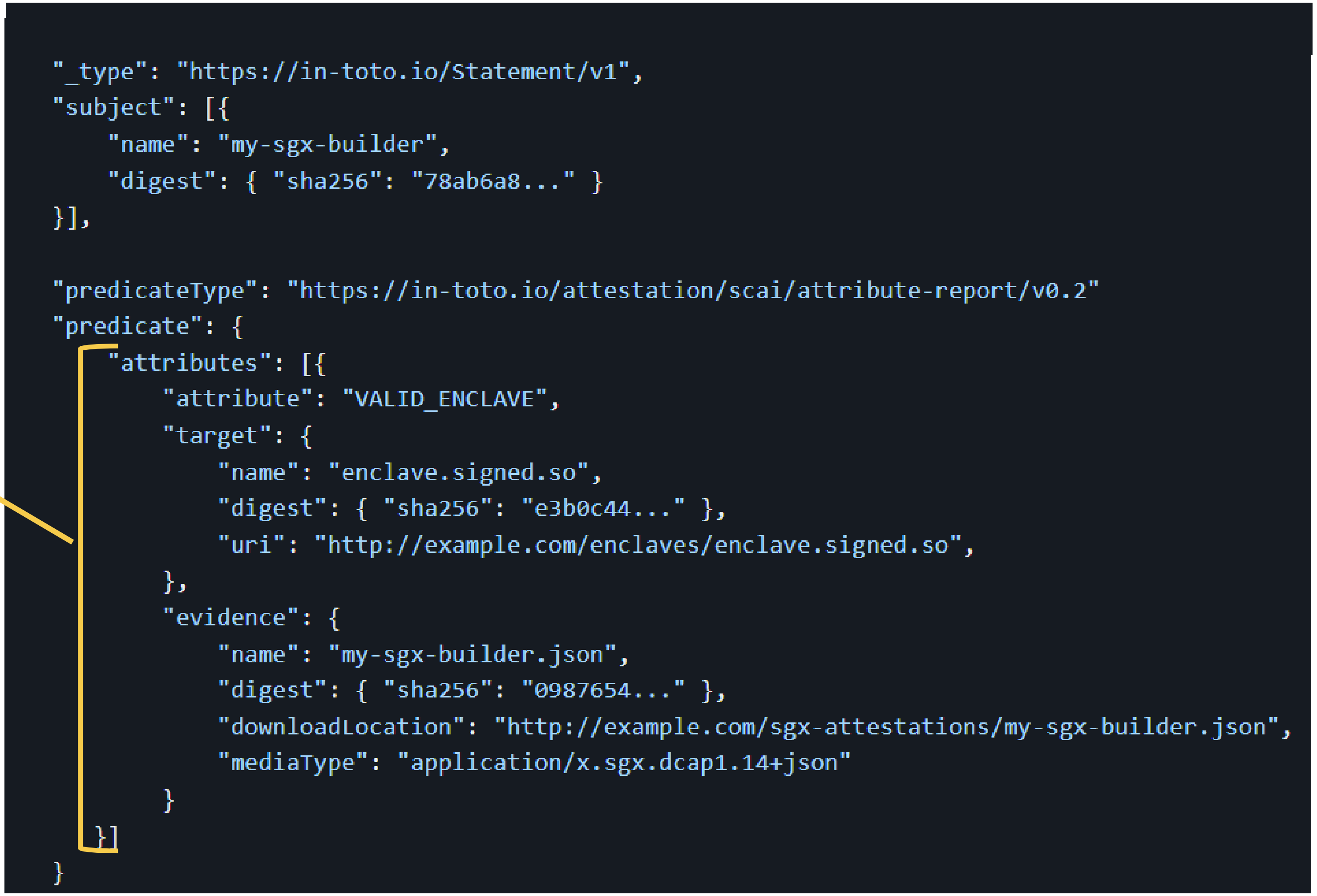
in-toto in a nutshell

in-toto Layout
(policy for SW
supply chain)

in-toto
Envelopes

Image sources: https://in-toto.io/; https://opencollective.com/bandit-sast; https://commons.wikimedia.org/wiki/File:Jenkins_logo.svg

Example: in-toto + SCAI Assertion for SGX enclave



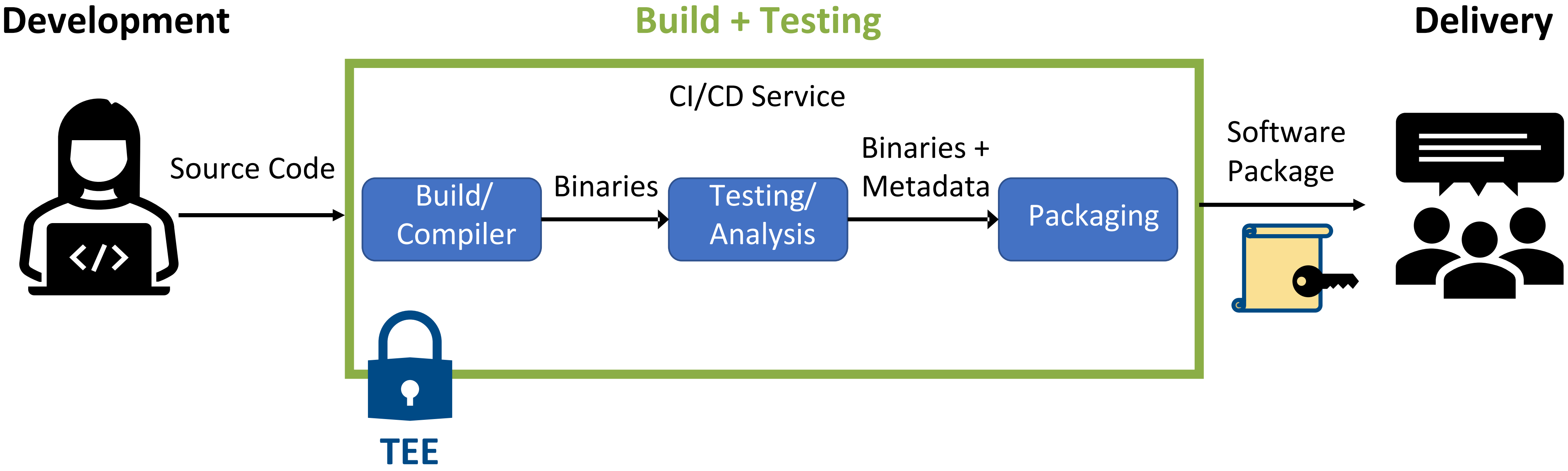
```
"_type": "https://in-toto.io/Statement/v1",
"subject": [{
    "name": "my-sgx-builder",
    "digest": { "sha256": "78ab6a8..." }
}],

"predicateType": "https://in-toto.io/attestation/scai/attribute-report/v0.2"
"predicate": {
    "attributes": [{
        "attribute": "VALID_ENCLAVE",
        "target": {
            "name": "enclave.signed.so",
            "digest": { "sha256": "e3b0c44..." },
            "uri": "http://example.com/enclaves/enclave.signed.so",
        },
        "evidence": {
            "name": "my-sgx-builder.json",
            "digest": { "sha256": "0987654..." },
            "downloadLocation": "http://example.com/sgx-attestations/my-sgx-builder.json",
            "mediaType": "application/x.sgx.dcap1.14+json"
        }
    }]
}
```

SCAI Predicate

Finally:

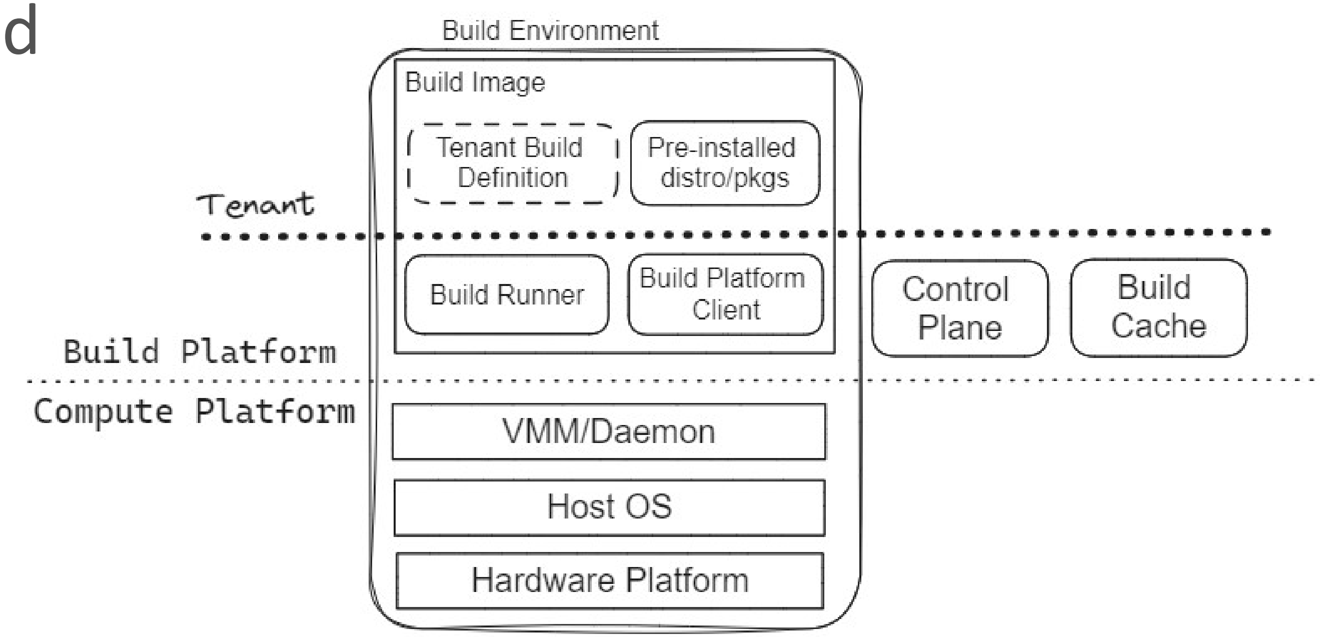
We need ***integrity*** for the SWSC process & metadata.

Example: TEE-enabled tools



Build Environment Attestations

- Goal: Use TEEs to attest to integrity of layers of the build environment's software stack
- Collaboration between Intel and GitHub via OpenSSF



OpenSSF = Open Source Security Foundation

Implementing TEE-enabled Builds

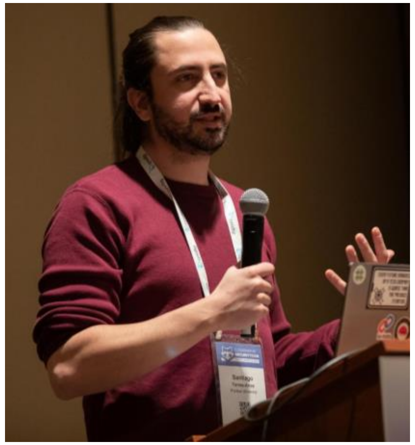
Level	Threat	Implementation requirements	Root of trust
L1	Tampering with build image distribution	Provide provenance of the build image (SW claims)	Build platform (e.g., GitHub, Google Build, GitLab)
L2	Tampering with build image kernel	Provision build image on secure boot-enabled platform	Compute platform (e.g., Msft Azure, GCP, AWS)
L3	Tampering with tenant build definition	Provision build image on run-time measured hardware	Hardware platform (e.g., Intel, AMD, ARM)

Build Environment Attestations: Status

- OpenSSF SLSA spec enhancement proposal in-flight
  - Set of integrity requirements for SW producers and build platforms
- Finalizing TEE-based requirements for build platforms
- Exploring how in-toto and SCAI can expose TEE attestations to SW-level consumers

SLSA = Supply-chain Levels for Software Artifacts
Pronounced “salsa”

Acknowledgements



Aditya Sirish A Yelgundhalli (NYU)
Tom Hennen (Google)
Parth Patel (Kusari)
Joshua Lock (Verizon)
Chad Kimes (GitHub)

…and many others!

Thanks! Questions?

Resources

SCAI Resources:

- SCAI @ in-toto: https://github.com/in-toto/attestation/blob/main/spec/predicates/scai.md
- Full SCAI Spec v0.1: https://arxiv.org/pdf/2210.05813.pdf