Applying COSE Signatures for YANG Data Provenance

draft-lopez-opsawg-yang-provenance-00

D. López (*Telefónica)*

IETF#117, San Francisco (US), July 2023

Prov·e·nance | ˈprävən(ə)ns |

- More specifically, *data provenance*
  - A documented trail accounting for the origin of a piece of data and where it has moved from to where it is presently
- Assurance of the origin and integrity of YANG datasets
  - Motivated by the discussion on metadata manifests
    - draft-ietf-opsawg- collected-data-manifest
  - Whenever the dataset is used beyond an original online flow
    - Use of data intermediaries, such as data lakes
    - AI/ML training and validation
    - Audit trails, including forensics evidence

The Foundations

- Current practice relies on the transport protocol
  - Identiy and crypto material in TLS, SSH…
  - Suitable for online flows
  - Contentious if used offline
- This proposal implies native support
  - Avoiding transitive trust
  - Very low impact on models using it
  - Recursion
- Based on COSE
  - Concise
  - Detached payload

Applying Provenance

- Add a leaf element containing a COSE signature
  - One and only one in the enclosing element
  - Anywhere

```
typedef provenance-signature {
      type binary;
      description
        "The provenance-signature type represents a digital signature
         associated to the enclosing element. The signature is based
         on COSE and generated using a canonicalized version of the
         enclosing element.";
      reference "draft-lopez-opsawg-yang-provenance";
}
```

```
module: ietf-platform-manifest
  +--ro platforms
  +--ro platform* [id]
  +--ro platform-provenance? provenance-signature
  +--ro id string
  +--ro name? string
  +--ro vendor? string
  +--ro vendor-pen? uint32
  +--ro software-version? string
  +--ro software-flavor? string
  +--ro os-version? string
  +--ro os-type? string
  +--ro yang-push-streams
  |  +--ro stream* [name]
  |     +--ro name
  |     +--ro description?
  +--ro yang-library
  + . . .
  .
  .
  .
```

Signatures

- COSE single signature string with *[nil]* payload
  - Algorithm-identifier, following COSE conventions and registries.
  - KID (Key ID), locally used and interpreted by the signer and the validator
  - The serialization method:
    - xml, json, cbor
  - Algorithm-parameters, following the COSE conventions
  - The signature, using as external supplied data
    - The whole element enclosing the signature leaf
    - Without the signature leaf element
    - Applying the corresponding canonicalization method

```
COSE_Sign1 = [
protected /algorithm-identifier, kid, serialization-method/
unprotected /algorithm-parameters/
signature /using as external data the content
           of the (meta-)data without the signature leaf/
]
```

What Comes Next

- Refining and detailing use cases
  - Proposals welcome
- Experimenting beyond the initial feasibility evaluation
  - Further COSE and serialization profiling
  - Detailed local KID rules
  - . . .
  - And a reference implementation
- Analyzing patterns for
  - Recursion: YANG nesting vs COSE multiple signatures
  - Addressing pipelining concepts
- And, for sure, seek for WG comments and support