Security

Michael McCool

30 June 2021

Outline

- New/Improved Security Data Localizers
  - body
  - uri
- Canonicalization
- Signing

Localizers

uri

- For when security data is embedded directly in the URI (ex: Phillips Hue)
- Use only when cannot be handled with "query" localizer
- Names URI template parameter
- Implied type is always "string"
- URI in associated forms need to include this template parameter
- Name conflicts to be avoided...

body

- For when security data is embedded in payload, possibly mixed in with other payload data, e.g. in a POST
- Uses a JSON Pointer to identify location in data schema where security data will be embedded
- Can refer to non-existent data schema element, *in which case it will be inserted*

Localizers – Example Schemes

uri

```
"securityDefinitions": {
  "uri_sec": {
      "scheme": "apikey",
      "in": "uri",
      "name": "KEY"
  }
},
"security": "uri_sec",
...
```

body

```
"securityDefinitions": {
  "body_sec": {
      "scheme": "apikey",
      "in": "body",
      "name": "/auth/key"
  }
},
"security": "body_sec",
...
```

Localizers – Example Forms

```
"color": { ...
   "type": "object",
   "properties": {
      "brightness": ...,
      "rgb": ...
  }
  "required": ["brightness", "rgb"],
     "forms": [{
        "href":
           "https://example.com/{KEY}/color",
        ...
     }]
}
```

```
"color": { ...
   "type": "object",
   "properties": {
      "brightness": ...,
      "rgb": ...,
      "auth": {
         "type": "object",
         "properties": {
            "key": { "type": "string" }
         },
        "required": ["key"]
        }
     },
     "required": ["brightness", "rgb", "auth"],
     "forms": [{
        "href": "https://example.com/color",
        ...
      }]
}
```

Localizers – Example Forms

```
"color": { ...
   "type": "object",
   "properties": {
      "brightness": ...,
      "rgb": ...
  }
  "required": ["brightness", "rgb"],
     "forms": [{
        "href":
            "https://example.com/{KEY}/color",
        ...
      }]
}
```

```
"color": { ...
   "type": "object",
   "properties": {
      "brightness": ...,
      "rgb": ...
   },
   "required": ["brightness", "rgb"],
   "forms": [{
       "href": "https://example.com/color",
       ...
    }]
}
```

Canonicalization

- Gives any unique information set a unique serialization
- JSON-LD canonical form in development; but will be based on RDF
- The TD canonical form needs only JSON processing
  - Based on JSON Canonical Serialization (RFC8785)
- Also need to deal with some special TD/JSON-LD features
  - Default values – always included, including for protocols
  - Named objects, e.g. security, data schemas – names must be retained
  - Prefixes – names must be retained, must be used if defined
  - Arrays/single values – omit [] for single values
  - Datetime – XML Canonical Form conventions
- Some of these impact RDF round-tripping, e.g. name preservation

Canonicalization Example

Original: Example 1 in TD spec

TD Canonical Form (plus White space): Example 37 in TD spec

TD Canonical Form (true; no white space): Example 38 in TD spec

Signing

- Goals:
  - Preserve integrity – prevent tampering, man-in-the-middle attacks
  - Verify source/originator
- Design
  - Follow structure of XML Signatures to extract subset of TD to sign
    - Multiple JSONPath/XPath queries
    - Result is hashed
  - Use JWS/JWA to actually compute signature on result of queries
  - Use JWK sets to reference keys (optional)
- Comments
  - Query support aligns with directory functionality
  - Needs to use "expanded canonical form" as input to queries
  - Still a PR: https://github.com/w3c/wot-thing-description/pull/1151

Signing - Example

```
"signatures": [{
   "signedInfo": [{
      "reference": "#/properties",
      "referenceType":
         "jsonpointer",
      "digest": "...",
      "digestAlg": "sha256"
   }],
   "sig": "...",
   "alg": "Ed448",
   "jku":
      "https://example.org/keys",
   "kid": "Key1",
}]
```

- One or more signatures; new ones added at the end to support chaining
- Each signature can have one or more signature info blocks, each with a query ("reference")
- Result of each query is hashed
- Signature info itself follows JWS/JWA spec (including recent elliptical curve extensions)
- Keys can optionally be referenced using JWK links