W3C Web of Things Security Definitions *Proposal*

Michael McCool

W3C WoT WG Security and Privacy TF

19 July 2018

Outline

- Review of current TD security metadata status
  - From metadata review presented at Bundang
- "Security Definitions" proposal
  - To address issues discussed in Bundang F2F
  - Designed as "backward-compatible" extension to current TD
- Discussion
  - Does proposal solve the identified problem?
  - Does it satisfy "least surprise"?
  - Is it feasible to implement in the JSON-LD1.1 framework?

Security Metadata Overview

- The "security" array can appear at
  - Thing level
  - Interaction level (top level of a property, action, or event)
  - Form level
  - Lower levels override higher levels - semantically, "pushed" to lowest level
- Value is an array of security configuration objects
  - JSON-LD 1.1: single value can be used in place of an array of one element
- Each security configuration has a
  - A "scheme" identifier (string value, conceptually from a fixed enumeration)
  - A set of parameters for that scheme
  - All schemes must be satisfied for access to be granted ("AND" logic)
  - "OR" logic is supported by using different configurations in parallel forms

Schemes

`basic`: username/password

`digest`: protected username/password

`apikey`: opaque value which allows access upon presentation

`bearer`: token which allows access upon presentation, internal structure provided by standard

`pop`: token which allows access after challenge-response so that stolen token does not provide access

`oauth2`: OAuth flows: allows use of FIDO flows to get tokens from authentication servers, use refresh servers, etc. Uses bearer tokens etc. as part of the flow.

Example: Proxy + Endpoint Configuration

```
"security": [
    {   "scheme": "digest",
        "proxyUrl": "https://portal.example.com/"
    },
    {   "scheme": "bearer",
        "format": "jwt",
        "alg": "ES256"
    }
]
```

Note: The "bearer" scheme does allow an authenticationUrl, but it is optional.

Notes

- Some schemes can be used with multiple protocols
  - In particular, "basic" can be applied to either HTTP or MQTT to represent basic username/password authentication
  - Applicability of a particular scheme to a particular protocol needs to be defined on a case-by-case basis
  - One challenge here is the required parameters may depend on the protocol…
- Special parameter "proxyUrl" can be used with all schemes
  - If used, indicates the configuration provides access to the proxy, not to the endpoint
  - Can use multiple schemes, even schemes of the same type, if one is for the proxy and one is for the endpoint
- Can also include a "description" with each security configuration

Extension Vocabulary Example: Payments

```
{"@context": ["https://w3c.github.io/wot/w3c-wot-td-context.jsonld",
              {"ilp": "http://interledger.org/"}],
...
"security": [
    {"scheme": "oauth2",…},
    {"scheme": "ilp:spsp",
     "ilp:account": "g.us.bank1.bob"
    }
]
```

To discuss: should ALL schemes be in external vocabularies associated with a protocol, eg `"scheme": "http:basic"`, etc.?

Issues Identified

- Redundancy
  - Even with current scheme (using layered overrides), redundancy can occur.
  - May need to repeat the same security configuration in different places, which is asking for trouble (eg configurations that should be the same not all getting updated)
  - Is especially troublesome if we want to attach human-readable documentation (eg "description" objects) to security configurations.
- Overrides are not always appropriate
  - It would be useful to have a way to "add" to the current set of configurations
- Mandatory "security" configuration desirable
  - When used with "scheme":"none" would make it clearer when there is no security on an object
  - May have to do this anyway if JSON-LD 1.1 can't distinguish empty array or empty object definition from no declaration

Proposal: Security Definitions

- New map object at the top level, "securityDefinitions"
- Map between strings and security configuration objects
  - It would be acceptable to use any value for these strings, including names used elsewhere or names of existing schemes.
- Strings could then be used in place of security configuration objects in "security" definitions.
  - Except for this extension to "security" objects, existing "security" definitions would not change
  - In particular, local security configurations would still be allowed, override mechanism would still be effective, etc.
- Security definitions do NOT imply usage of a security configuration; they are just definitions, not bindings.

Example 1: Security Definitions

```
"securityDefinitions": {
    "proxydigest": {
        "scheme": "digest",
        "proxyUrl": "https://portal.example.com/"
    },
    "bearer": {
        "scheme": "bearer",
        "format": "jwt",
        "alg": "ES256"
    }
}
```

Example 1: Extended Security Binding

Later, inside a form:

```
"security": [
    "proxydigest",
    { "scheme": "basic" }
]
```

Example 1: Equivalent Security Binding

```
"security": [
    {
        "scheme": "digest",
        "proxyUrl": "https://portal.example.com/"
    },
    {
        "scheme": "basic"
    }
]
```

Semantic equivalence: the "proxydigest" string acts as if it were replaced by its defined value.

Example 2: High-Level Definition + Binding

```
"securityDefinitions": {
    "proxydigest": {
        "scheme": "digest",
        "proxyUrl": "https://portal.example.com/"
    },
    "bearer": {
        "scheme": "bearer",
        "format": "jwt",
        "alg": "ES256"
    }
},
"security": ["proxydigest","bearer"],
```

Example 3: Single-Value Case

```
"securityDefinitions": {
    "auth": {
        "scheme": "oauth2",
        "description": …
    }
},
"security": "auth"
```

Empty and Mandatory Security Schemes

- Issues:
  - Having "no security" by default seems to set a bad precedent
  - Currently, using "empty array or configuration" to mean "no security" may cause semantics issues (empty nodes, etc) and it's not clear if an "empty" security declaration can be distinguished from no declaration.
- Proposal:
  - Additional scheme "none" to indicate no security
  - Make "security" tag mandatory
  - Can still use overrides to avoid including "security" tag in each form (eg "mandatory" means it has to occur at some level above the current object.)
- Discussion:
  - Can the "security tag is mandatory at some level the current object" rule be enforced/validated with current tools?