Network Working Group C. Bormann
Internet-Draft Universität Bremen TZI
Intended status: Informational 28 June 2020
Expires: 30 December 2020

An Authorization Information Format (AIF) for ACE
draft-bormann-core-ace-aif-09

Abstract

Constrained Devices as they are used in the "Internet of Things" need security. One important element of this security is that devices in the Internet of Things need to be able to decide which operations requested of them should be considered authorized, need to ascertain that the authorization to request the operation does apply to the actual requester, and need to ascertain that other devices they place requests on are the ones they intended.

To transfer detailed authorization information from an authorization manager (such as an ACE-OAuth Authorization Server) to a device, a representation format is needed. This document provides a suggestion for such a format, the Authorization Information Format (AIF). AIF is defined both as a general structure that can be used for many different applications and as a specific refinement that describes REST resources and the permissions on them.

Table of Contents

1. Introduction

(See Abstract.)

1.1. Terminology

This memo uses terms from [RFC7252] and [RFC4949].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here. These words may also appear in this

document in lower case as plain English words, absent their normative meanings.

(Note that this document is itself informational, but it is discussing normative statements that MUST be put into concrete terms in each specification that makes use of this document.)

The term "byte", abbreviated by "B", is used in its now customary sense as a synonym for "octet".

2. Information Model

Authorizations are generally expressed through some data structures that are cryptographically secured (or transmitted in a secure way). This section discusses the information model underlying the payload of that data (as opposed to the cryptographic armor around it).

For the purposes of this strawman, the underlying access control model will be that of an access matrix, which gives a set of permissions for each possible combination of a subject and an object. We do not concern the AIF format with the subject for which the AIF object is issued, focusing the AIF object on a single row in the access matrix (such a row traditionally is also called a capability list). As a consequence, AIF MUST be used in a way that the subject of the authorizations is unambiguously identified (e.g., as part of the armor around it).

The generic model of a such a capability list is a list of pairs of object identifiers and the permissions the subject has on the object(s) identified.

```
AIF-Generic<Toid, Tperm> = [* [Toid, Tperm]]
```

Figure 1: Definition of Generic AIF

In a specific data model, the object identifier ("Toid") will often be a text string, and the set of permissions ("Tperm") will be represented by a bitset in turn represented as a number (see Section 3).

```
AIF-Specific = AIF-Generic<tstr, uint>
```

Figure 2: Likely shape of a specific AIF

2.1. REST-specific model

In the specific instantiation of the REST resources and the permissions on them, for the object identifiers ("Toid"), we simply use the URI of a resource on a CoAP server. More specifically, the parts of the URI that identify the server ("authority" in [RFC3986]) are considered the realm of the authentication mechanism (which are handled in the cryptographic armor); we therefore focus on the "path-absolute" and "query" parts of the URI (URI "local-part" in this specification, as expressed by the Uri-Path and Uri-Query options in CoAP). As a consequence, AIF MUST be used in a way that it is unambiguous who is the target (enforcement point) of these authorizations.

For the permissions ("Tperm"), we simplify the model permissions to giving the subset of the CoAP methods permitted. This model is summarized in Table 1.

local-part	Permission Set
/s/light	GET
/a/led	PUT, GET
/dtls	POST

Table 1: An authorization

instance in the AIF
Information Model

2.2. Limitations

This simple information model only allows granting permissions for statically identifiable objects, e.g. URIs for the REST-specific instantiation. One might be tempted to extend the model towards URI templates [RFC6570], however, that requires some considerations of the ease and unambiguity of matching a given URI against a set of templates in an AIF object.

This simple information model also doesn't allow further conditionalizing access based on state outside the identification of objects (e.g., "opening a door is allowed if that isn't locked").

Finally, the model does not provide any special access for a set of resources that are specific to a subject, e.g. that the subject created itself by previous operations (PUT, POST) or that were specifically created for the subject by others.

2.3. Extended REST-specific model

The extended REST-specific model addresses the need to provide defined access to dynamic resources that were created by the subject itself, specifically, a resource that is made known to the subject by providing Location-* options in a CoAP result or using the Location header field in HTTP [RFC7231] (the Location-indicating mechanisms). (The concept is somewhat comparable to "ACL inheritance" in NFSv4 [rfc5661], except that it does not use a containment relationship but the fact that the dynamic resource was created from a resource to which the subject had access.)

local-part	Permission Set
/a/make-coffee	POST, Dynamic-GET, Dynamic-DELETE

Table 2: An authorization instance in the AIF
Information Model

For a method X, the presence of a Dynamic-X permission means that the subject holds permission to exercise the method X on resources that have been returned by a Location-indicating mechanism to a request that the subject made to the resource listed ("/a/make-coffee" in the example, which might return the location of a resource that allows GET to find out about the status and DELETE to cancel the coffee-making operation).

Since the use of the extension defined in this section can be detected by the mentioning of the Dynamic-X permissions, there is no need for another explicit switch between the basic and the extended model; the extended model is always presumed once a Dynamic-X permission is present.

3. Data Model

Different data model specializations can be defined for the generic information model given above.

In this section, we will give the data model for basic REST authorization. As discussed, the object identifier is specialized as a text string giving a relative URI (local-part as absolute path on the server serving as enforcement point). The permission set is specialized to a single number by the following steps:

* The entries in the table that specify the same local-part are merged into a single entry that specifies the union of the permission sets.

* The (non-dynamic) methods in the permission sets are converted into their CoAP method numbers, minus 1.

* Dynamic-X permissions are converted into what the number would have been for X, plus a Dynamic-Offset chosen as 32 (e.g., 35 for

Dynamic-DELETE).

* The set of numbers is converted into a single number by taking each number to the power of two and computing the inclusive OR of the binary representations of all the power values.

This data model could be interchanged in the JSON [RFC8259] representation given in Figure 3.

```
[["/s/light", 1], ["/a/led", 5], ["/dtls", 2]]
```

Figure 3: An authorization instance encoded in JSON (46 bytes)

In CDDL [RFC8610], a straightforward specification of the data model (including both the methods from [RFC7252] and the new ones from [RFC8132], identified by the method code minus 1) is:


```
AIF-REST = AIF-Generic<path, permissions>
path = tstr   ; URI relative to enforcement point
permissions = uint .bits methods
methods = &(
  GET: 0
  POST: 1
  PUT: 2
  DELETE: 3
```

```
     FETCH: 4
     PATCH: 5
     iPATCH: 6
     Dynamic-GET: 32; 0 .plus Dynamic-Offset
     Dynamic-POST: 33; 1 .plus Dynamic-Offset
     Dynamic-PUT: 34; 2 .plus Dynamic-Offset
     Dynamic-DELETE: 35; 3 .plus Dynamic-Offset
     Dynamic-FETCH: 36; 4 .plus Dynamic-Offset
     Dynamic-PATCH: 37; 5 .plus Dynamic-Offset
     Dynamic-iPATCH: 38; 6 .plus Dynamic-Offset
   )
```

Figure 4: AIF in CDDL

A representation of this information in CBOR [RFC7049] is given in Figure 5; again, several optimizations/improvements are possible.

```
   83                        # array(3)
      82                     # array(2)
         68                  # text(8)
            2f732f6c69676874 # "/s/light"
         01                  # unsigned(1)
      82                     # array(2)
         66                  # text(6)
            2f612f6c6564     # "/a/led"
         05                  # unsigned(5)
      82                     # array(2)
         65                  # text(5)
            2f64746c73       # "/dtls"
         02                  # unsigned(2)
```

Figure 5: An authorization instance encoded in CBOR (29 bytes)

Note that choosing 32 as Dynamic-Offset means that all future CoAP methods that can be registered can be represented both as themselves and in the Dynamic-X variant, but only the dynamic forms of methods 1 to 21 are typically usable in a JSON form [RFC7493].

4. Media Types

This specification defines media types for the generic information model, expressed in JSON ("application/aif+json") or in CBOR ("application/aif+cbor"). These media types have parameters for specifying "Toid" and "Tperm"; default values are the values "local-uri" for "Toid" and "REST-method-set" for "Tperm".

[Insert lots of boilerplate here]

A specification that wants to use Generic AIF with different "Toid" and/or "Tperm" is expected to request these as media type parameters (Section 5.2) and register a corresponding Content-Format (Section 5.3).

5. IANA Considerations

5.1. Media Types

See Section 4.

5.2. Registries

IANA is requested to create a registry for AIF with two sub-registries for "Toid" and "Tperm", populated with:

Subregistry	name	Description/Specification
Toid	local-part	local-part of URI as specified in [RFCthis]
Tperm	REST-method-set	set of REST methods represented as specified in [RFCthis]

Table 3

The registration policy is Specification required [RFC8126]. The designated expert will engage with the submitter to ascertain the requirements of this document are addressed.

5.3. Content-Format

IANA is requested to register Content-Format numbers in the CoRE Parameters Registry [IANA.core-parameters], as follows:

6. Security Considerations

(TBD. Some issues are already discussed in the security considerations of [RFC7252] and in [RFC8576].)

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

7.2. Informative References

[I-D.ietf-ace-dtls-authorize]
Gerdes, S., Bergmann, O., Bormann, C., Selander, G., and L. Seitz, "Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE)", Work in Progress, Internet-Draft, draft-ietf-ace-dtls-authorize-11, 18 June

2020, <http://www.ietf.org/internet-drafts/draft-ietf-ace-dtls-authorize-11.txt>.

[I-D.ietf-ace-oscore-profile]
Palombini, F., Seitz, L., Selander, G., and M. Gunnarsson, "OSCORE profile of the Authentication and Authorization for Constrained Environments Framework", Work in Progress, Internet-Draft, draft-ietf-ace-oscore-profile-11, 18 June 2020, <http://www.ietf.org/internet-drafts/draft-ietf-ace-oscore-profile-11.txt>.

[IANA.core-parameters]
IANA, "Constrained RESTful Environments (CoRE) Parameters", <http://www.iana.org/assignments/core-parameters>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[rfc5661] Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, DOI 10.17487/RFC5661, January 2010, <https://www.rfc-editor.org/info/rfc5661>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <https://www.rfc-editor.org/info/rfc6570>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015,

<https://www.rfc-editor.org/info/rfc7493>.

[RFC8132] van der Stok, P., Bormann, C., and A. Sehgal, "PATCH and FETCH Methods for the Constrained Application Protocol (CoAP)", RFC 8132, DOI 10.17487/RFC8132, April 2017, <https://www.rfc-editor.org/info/rfc8132>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8576] Garcia-Morchon, O., Kumar, S., and M. Sethi, "Internet of Things (IoT) Security: State of the Art and Challenges", RFC 8576, DOI 10.17487/RFC8576, April 2019, <https://www.rfc-editor.org/info/rfc8576>.

Acknowledgements

Jim Schaad and Francesca Palombini provided comments that shaped the direction of this document.

Author's Address

Carsten Bormann
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org