OCF 1.0 Security CR – Bug 1457 (Bug 1440)

Legal Disclaimer

THIS IS A DRAFT SPECIFICATION DOCUMENT ONLY AND HAS NOT BEEN ADOPTED BY THE OPEN CONNECTIVITY FOUNDATION. THIS DRAFT DOCUMENT MAY NOT BE RELIED UPON FOR ANY PURPOSE OTHER THAN REVIEW OF THE CURRENT STATE OF THE DEVELOPMENT OF THIS DRAFT DOCUMENT. THE OPEN CONNECTIVITY FOUNDATION AND ITS MEMBERS RESERVE THE RIGHT WITHOUT NOTICE TO YOU TO CHANGE ANY OR ALL PORTIONS HEREOF, DELETE PORTIONS HEREOF, MAKE ADDITIONS HERETO, DISCARD THIS DRAFT DOCUMENT IN ITS ENTIRETY OR OTHERWISE MODIFY THIS DRAFT DOCUMENT AT ANY TIME. YOU SHOULD NOT AND MAY NOT RELY UPON THIS DRAFT DOCUMENT IN ANY WAY, INCLUDING BUT NOT LIMITED TO THE DEVELOPMENT OF ANY PRODUCTS OR SERVICES. IMPLEMENTATION OF THIS DRAFT DOCUMENT IS DONE AT YOUR OWN RISK AMEND AND IT IS NOT SUBJECT TO ANY LICENSING GRANTS OR COMMITMENTS UNDER THE OPEN CONNECTIVITY FOUNDATION INTELLECTUAL PROPERTY RIGHTS POLICY OR OTHERWISE. IN CONSIDERATION OF THE OPEN CONNECTIVITY FOUNDATION GRANTING YOU ACCESS TO THIS DRAFT DOCUMENT, YOU DO HEREBY WAIVE ANY AND ALL CLAIMS ASSOCIATED HEREWITH INCLUDING BUT NOT LIMITED TO THOSE CLAIMS DISCUSSED BELOW, AS WELL AS CLAIMS OF DETRIMENTAL RELIANCE.

1 Background

The core architecture specification defined a parameter for security that is exercised when the application or vendor instantiates a resource – commonly referred to as the “sec” flag. This flag indicates to the core framework to apply a stronger security policy when set to true. IoTivity ensures resources must be accessed through DTLS (COAPS) when “sec” equals true. When “sec” equals false either COAP or COAPS may be used – essentially the requestor decides whether stronger security is needed.

However, the security specification layers additional security functionality in the form of an ACL policy that matches “subjects” in terms of their device ID, role and wildcard grouping semantics on top of “sec” flag semantics. This has resulted in redundancy while still leaving coverage gaps when considering possible legitimate uses of COAPS.

The framework may distinguish between the following 4 ways in which a client may choose to connect to the server:

1. Subject arrives over an authenticated and encrypted channel.
2. Subject arrives over an authenticated and unencrypted channel.
3. Subject arrives over an anonymous and encrypted channel.
4. Subject arrives over an anonymous and unencrypted channel.

Items 1 and 4 are currently supported by the OCFv1.0 specification using CoAP and CoAPS (with defined DTLS ciphersuites). Items 2 and 3 could be supported by a vendor but currently these are not mandated by an OCF specification.

Items 2 and 3 require DTLS ciphersuite definition and incorporation into the specification. It may be possible that vendor-specific implementations achieve 2 or 3.

The OICv1.1 specification identifies a wildcard ‘*’ that applies to the /oic/r/acl resource that matches any subject whether or not they are authenticated.

The Security Working group has reviewed the following wildcard grouping semantics and determined that they may be useful:

Case 1: Any subject (authenticated or not; encrypted or not) can access the resource.

Case 2: Any subject who is authenciated (encrypted or not) can access the resource.

Case 3: Any subject that establishes an encrypted channel (authenticated or not) can access the resource.

Case 4: Any subject who is authenticated and not encrypted can access the resource.

A truth table can be constructed to identify the 4 possible ways in which a subject may connect to a server where “A” means the subject is authenticated and “E” means the channel / message is encrypted. (It is assumed the channel / message is integrity protected). !A means NOT authenticated (aka anonymous) and !E means not encrypted.

1. (A, E)
2. (A, !E) – Not supported currently by OCF
3. (!A, E) – Not supported currently by OCF
4. (!A, !E)

A goal of access control policy authoring is to identify subject groupings that meaningfully map to a set of permissions on resources to achieve a desirable security objective. For example, resources involved in the performance of a control objective likely require stronger authentication while resources that return sensitive data likely require stronger encryption.

CRUDN permissions can roughly be divided accordingly where C-UD- may be classified as ‘control’ related and –R—N as sensing or data producing. Truth table items (1) and (2) may be most appropriately aligned with permissions for doing control while items (1) and (3) may be most appropriate for sensing or analytics. Item (4) may be necessary for discovery but care should be taken to ensure discovery doesn’t undermine the other goals. This is achieved by carefully considering which resources and properties are sufficient for achieving a particular discovery related outcome.

OCF interfaces may play a role when defining access semantics that are tied to subject groups. For example, the oic.if.s interface identifies resources whos primary function is to return data, while oic.if.a interface identifies used for control. It may be appropriate to align the (1) and (3) subject groups with a resource matching policy specified in terms of oic.r.s and similarly for (1) and (2) matching in terms of the oic.r.a interface. Other interfaces combine both such as oic.if.rw and oic.if.baseline, hence it may be appropriate to specify (1) subject group in a separate ACE that includes read-write interface classes.

It may be observed that while some degree of subject grouping is beneficial, it is possible to construct subject grouping semantics that are too flexible resulting in uninteresting permissions and resource matching options.

The subject grouping proposed to the SWG may be an example of too much flexibility. Consider the Venn diagram of truth table elements for each wildcard case.

- OCF_ANY_REQUESTER implies [(A,E) OR (!A,E) OR (A,!E) OR (!A,!E)]
- OCF_ANY_AUTHENTICATED_REQUESTER_WITH_ENCRYPTION implies [(A,E)]
- OCF_ANY_REQUESTER_WITH_ENCRYPTION implies [(A,!E) OR (!A,!E)]

- OCF_ANY_AUTHENTICATED_REQUESTER implies [(A,E) OR (A,!E)]

The Venn diagram shows (A, E) overlapping all groups implying the permissions and resource matching must not be specific to just (A, E), but must consider the least common access given the other members of the grouping. While it is possible to separate out (A, E) as it is in a group by itself, it isn't possible to separate out the other tuple parings. In particular, the (!A, !E) tuple is particularly interesting from the perspective that increased care should be given to consider the security and privacy impact for resources exposed with not protections. While it may be correct to observe that if a resource is allowed for (!A, !E), accessing it with stronger security contexts doesn't present a security error. The potential for policy authors to become confused about which group the policy should target is increased if multiple levels of security are combined as part of the group definition. Given the ability to broadly specify resource matching using wildcards, the added user convenience of mixing subject groupings seems minimal. That combined with an expectation that CRUDN permissions should not grant excess privilege, there will naturally emerge three or four ACEs aligned around the truth table items.

2 Overview

This CR defines ACE subject wildcard matching based on a 2-tuple structure matching the above truth table.

This CR applies only to oic.sec.ace2 and version checking is required when working with OICv1.1 that defines the oic.sec.ace resource – which is unchanged in OCFv1.0.

This CR defines two cases {"A"=true, "E"=true} and {"A"=false, "E"=false} as OCFv1.0 has the facilities to implement only these two. When (if) OCF specifications provide facilties for {"A"=false, "E"=true} and {"A"=true, "E"=false}; the subject ACE matching specification may be revisited to incorporate these cases into ACL enforcement.

If an OCFv1.0 ACE is authored to include unsupported tuples, the server shall return an error in response to an UPDATE message that contains unsupported subjects. The error code shall be ACCESS_DENIED_NO_ACE.

3 ACE2 Subject Matching Affected Sections

This applies to Section 5.1.

Lines 396-400:

"If access control is SBAC, then there needs to be an ACE for each subject (identity of an OIC client) that needs to access a SBAC controlled resource. However, ACLs for unknown or anonymous (unauthenticated) subject may be possible and subject to default permissions defined for the resource. For example:

```
Example ACL: uuid:0000-0000-0000-0000 -> "/oic/*" ? 0x01 (read-only)"
```

Becomes:

“The ACE must match both the subject (i.e. OCF Client) and the resource requested for the ACE to apply. There are multiple ways a subject could be matched, (1) device id, (2) role or (3) wildcard. The way in which the client connects to the server may be relevant context for making access control decisions. Wildcard matching on authenticated vs. unauthenticated and encrypted vs. unencrypted connection allows an access policy to be broadly applied to subject classes.”

Example Wildcard Matching Policy:

```
“aces”: [
  {
      “subject”: {“conntype” : “anon-clear” },
      “resources”:[
          { “rule”:“*” }
        ],
      “permission”: 31
  },
  {
      “subject”: {“conntype” : “auth-crypt” },
      “resources”:[
          { “rule”:“*” }
        ],
      “permission”: 31
  },
]
```

This applies to Section 5.1.1.

Lines 412 – 416:

“When an OIC Client device requests access to resources from an OIC Server,the OIC Server examines the OIC client’s access rights to its resources based SBAC or RBAC. Access requests may be authorized based on group or device credentials. The ACL architecture illustrates four client devices seeking access to server resources. A server evaluates each request using local ACL policies and Access Manager Service.”

Becomes:

“The server examines the resource(s) requested by the client before processing the request. The access control resources (e.g. /oic/r/acl, /oic/r/acl2, etc…) are searched to find one or more ACE entries that match the requestor and the requested resources. If a match is found then permission and period constraints are applied. If more than one match is found then the logical UNION of permissions is applied to the overlapping periods.

The server uses the connection context to determine whether the subject has authenticated or not and whether data confidentiality has been applied or not. Subject matching wildcard policies can match on each aspect. If the user has

authenticated, then subject matching may happen at increased granularity based on role or device identity.“

This applies to Section 6.1.

Line 779:

“"Subject": "*",”

Becomes:

```
“Subject”: {“conntype” : “anon-clear”},”
```

This applies to Section 13.

Table 23:

The box labelled oic.sec.ace Property should be replaced with a diagram that shows the new oic.sec.subject structure defined below.

This applies to CR48 and Section 13.5.2.

Definition of “oic.sec.subject” to be:

```
“anyof” : [
          “property”: “oic.sec.roletype”,
          “property”: “oic.sec.didtype”,
          “property”: “oic.sec.conntype”
        ]
where oic.sec.conntype is:

"conntype": {
        "type": "string",
        "enum": [ "auth-crypt", "anon-clear" ],
        "description": "This property allows an ACE to be matched based
on the connection or message protection type",
        "detail-desc": [
            "auth-crypt - ACE applies if the Client is authenticated
and the data channel or message is encrypted and integrity protected",
            "anon-clear - ACE applies if the Client is not
authenticated and the data channel or message is not encrypted but may
be integrity protected"
        ]
     }
```

3.1 Host Reference Matching

Examples: JSON for subject wildcard matching

```
//matches all subjects that have authenticated and confidentiality
protections in place.
“subject” : {
     “conntype” : “auth-crypt”
}
```

```
//matches all subjects that have NOT authenticated and have NO
confidentiality protections in place.
“subject” : {
    “conntype” : “anon-clear”
}
```

13.5 ACL Resources(/oic/sec/acl)

13.5.2 ACL Resource

The **oic.sec.ace2** structure is defined as follows:

Property Name	Value Type	Mandatory	Description
subject	anyof : oic.sec.roletype, oic.sec.didtype, oic.sec.**conntype**	Yes	The OCF Client is the subject of the ACE when the roles, device identity, or connection type matches.
resources	array of oic.sec.ace2.resource-ref	Yes	The application’s resources to which a security policy applies
permission	oic.sec.crudntype.bitmask	Yes	Bitmask encoding of CRUDN permission
validity	array of oic.sec.time-pattern	No	An array of a tuple of period and recurrence. Each item in this array contains a string representing a period using the RFC5545 Period, and a string array representing a recurrence rule using the RFC5545 Recurrence.
aceid	integer	No	~~Aceid is unique with respect to the ‘aces’ array. It is used as part of a query string that identifies which array record to change when using the UPDATE command.~~ ~~If the aceid is not supplied a new record is appended to the array and the server provides a value that does not conflict with existing values.~~ ~~DELETE shall provide aceid to update a single record otherwise the entire array will be deleted.~~ Note: Behavior is defined in BZ1485

Table 32 – oic.sec.ace2 data type definition.