Admin Interface for the OSCORE Group Manager

draft-ietf-ace-oscore-gm-admin-05

Marco Tiloca, RISE
Rikard Höglund, RISE
Peter van der Stok
Francesca Palombini, Ericsson

IETF 113, ACE WG, March 22\textsuperscript{nd}, 2022
Recap

› RESTful admin interface at the OSCORE Group Manager
  – Create, (re-)configure and delete OSCORE groups
  – Support for both: i) Link Format and CBOR ; ii) CoRAL

› Two new types of resources at the Group Manager
  – A single group-collection resource, at /manage
  – One group-configuration resource per group, at /manage/GROUPNAME

› Using ACE for authentication and authorization
  – The Administrator is the Client
  – The Group Manager is the Resource Server
  – For secure communication, use transport profiles of ACE
Overview

**Group-collection resource**
- Retrieve the list of OSCORE groups
  - All groups (GET)
  - Group selected by filters (FETCH)
- Create a new OSCORE group (POST)
  - A group-configuration resource is created
  - A group-membership for joining nodes is also created, see *ace-key-groupcomm-oscore*

**Configuration Properties**
- Security algorithms and parameters

**Status Properties**
- Any other information (e.g., group name)

**Group-configuration resource**
- Retrieve the group configuration (GET)
- Retrieve part of the group configuration (FETCH)
- Overwrite the group configuration (PUT)
- Update the group configuration (PATCH/iPATCH)
- Delete the group (DELETE)

Figure 1: Resources of a Group Manager
Updates since IETF 112

› **Terminology update**
  – Triggered by the revision of *draft-ietf-core-oscore-groupcomm*
  – Clear distinction between “public key” and “authentication credential”
  – Renamed the parameter ‘pub_key_enc’ to ‘cred_fmt’

› **Simplified selection/negotiation of group name upon group creation**
  – **Kept**: the name actually assigned to the new group is a decision of the Group Manager
  – **Kept**: the assigned group name has to be available at the Group Manager
  – **Updated**: the Administrator creating the group **has to** provide a suggested name
  – **Updated**: if the suggested name is already taken, the Group Manager assigns an available one
    ‣ Keep the assignment of group names flexible and ultimately up to the Group Manager
    ‣ Keep a tractable checking of group creation requests against authorization information in the token (more on this later)
Updates since IETF 112

- **Updates of existing group configuration (PUT/PATCH/iPATCH)**
  - Now made explicit how to inform current group members of the new configuration
  - Send a subset of the “Joining Response” message defined in `draft-ace-key-groupcomm-oscore`
  - Use the same content format `application/ace-groupcomm+cbor`

- **Considered possible addition upon group creation**
  - The Group Manager may recycle OSCORE Group IDs in a group
    - This allows an OSCORE group to “live forever”
    - Recently changed to be an *optional* feature in `draft-ietf-core-oscore-groupcomm`
  - **Should the Administrator have any saying in this when creating a group?** Proposal:
    - Define a new parameter for the group creation request, to indicate a group Status Property
    - If “true”, the Group Manager recycles Group IDs if actually able to
    - This cannot be changed later on as part of a group configuration update

\[ Ok \; to \; add? \]
Updates since IETF 112

- Defined a proper format of ‘scope’, using an AIF data model
  - Driven mostly by two discussions

- Early comment from Jim Schaad
  - An Administrator uploads a token T1 at the Group Manager
  - The Administrator creates groups G1 and G2
  - T1 expires; the Administrator gets a new token T2 and uploads it at the Group Manager
  - The Administrator has a new identity ➔ Not recognizable as the creator of G1 and G2!
  - What should ‘scope’ be in token T2, such that:
    - The Administrator can create new groups, and continue accessing G1 and G2
      - Not trivial: the Group Manager took the final decision on G1 and G2 names
    - There is no need to update access policies on the Authorization Server
Updates since IETF 112

› **More comments from Christian Amsüss**
  – Good to admit multiple Administrators for a same group, with different privileges
  ‹ A set of Administrators can access an existing group configuration resource, ...  
  ‹ ... as allowed to perform some operations on a group created by another Administrator
  – This opens to “classes” of Administrators, to be enforced through ‘scope’

› **Follow-up discussions among co-authors led to ...**
  – ... what was in Section 2.1.1 of v-04 as a placeholder, with a technical direction ...
  – ... which is now fully elaborated in the latest v-05
Use a structured scope and AIF

† How is scope in ace-key-groupcomm-oscore?
  † This is for users of groups
    † Group members; external signature verifiers
  † Using the AIF-OSCORE-GROUPCOMM data model
  † Good to consider as a starting point

† Scope = << [+ scope_entry ] >>
  † scope_entry = [Toid, Tperm]
  † Toid : tstr, with value a group name
  † Tperm : uint, encoding roles as flag bits

<table>
<thead>
<tr>
<th>bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role allowed if bit set to 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0: Reserved (bit always set to 0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: Requester</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: Responder</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3: Monitor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4: Verifier</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5, 6, 7: Unassigned</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
New AIF Data Model – AIF-Generic<Toid, Tperm> = [ *[Toid, Tperm] ]

- Toid: Text string, specifying a wildcard pattern for group names
- Tperm: Unsigned integer, indicating admin permissions as flag bits
- Permissions apply to groups whose name matches the pattern!

Possible permissions in Tperm

- 0: Retrieve list of existing security groups
  - Always granted
- 1: Create a new group and its configuration
- 2: Read the configuration of a group
- 3: Overwrite/update a group configuration
- 4: Delete a group and its configuration

Permissions are related to a name pattern
- They survive across different issued tokens and changes of security identity (Jim’s point)

Possible to consider more Administrators than the group creator (Christian’s point)
- Expected for a creator: (1)(2)(3)(4) all granted
- Expected for a non-creator: (1) not granted; some of (2)(3)(4) granted; restrictive name pattern
Format of ‘scope’ in gm-admin (2/3)

New data model AIF-OSCORE-GROUPCOMM-ADMIN
- This is for Administrators of groups
- Admit creator and non-creator Administrators

Scope = << [ + scope_entry ] >>
- scope_entry = [Toid, Tperm]
  - Toid : tstr, i.e., a wildcard pattern of group names
  - Tperm : uint, encoding permissions as bit flags
    Permissions apply to groups whose name matches the pattern in Toid!

Any comments?
Format of ‘scope’ in gm-admin (3/3)

What does it mean on the Group Manager as Resource Server? (Section 6)
- An Administrator request is served if ‘scope’ has at least one scope entry allowing so
- Added detailed rules for request processing to each resource handler

What does it mean on the Authorization Server? (Section 4)
- As usual, check the requested ‘scope’ against access policies for the Administrator
  - If not possible to grant as is, grant the intersection of what is asked and what is allowed
  - Practically, this gets tricky when checking name patterns against name patterns
- The current text has an actionable and very detailed procedure for the AS
- Proposal for next version:
  - Keep the high level process and goal above in Section 4
  - Move the detailed procedure to an Appendix, as an example

Objections?
Todo (?): mixed set of scope entries

- Under a same Group Manager a Client might be both:
  - (A) User for some groups
  - (B) Administrator for some groups

- The two types of scope entry are distinguishable!
  - For A, the least significant bit is always 0
  - For B, the least significant bit is always 1

- Proposal: allow both types of scope entry to be present in the same scope

- Objections?
Summary and next steps

› Latest updates
  – Terminology and parameters consistent with “public key” vs. “authentication credential”
  – Defined AIF data model to express ‘scope’ for Administrators
  – Updated request processing at the Group Manager, per the AIF-based authorization info
  – Simplified selection/negotiation of group name upon group creation
  – Revised order of content in Sections 2-5; editorial improvements

› Planned next steps
  – Consider allowing ‘scope’ to include a mix of:
    › Scope entries for Administrators (AIF data model defined here)
    › Scope entries for group users (AIF data model from ace-key-groupcomm-oscore)
  – Consider moving detailed scope checking procedure at the AS to an appendix
  – More details on error handling (e.g., no group names currently available to assign)

› Comments and reviews are welcome!
Thank you!

Comments/questions?

https://github.com/ace-wg/ace-oscore-gm-admin
Backup
Group Configuration Parameters

› Configuration properties
  – hkdf
  – cred_fmt
  – group_mode
  – sign_enc_alg
  – sign_alg
  – sign_params
  – pairwise_mode
  – alg
  – ecdh_alg
  – ecdh_params
  – det_req
  – det_hash_alg

› Status properties
  – rt = “core.osc.gconf”
  – active
  – group_name  // Plain immutable identifier
  – group_title  // Descriptive string
  – ace_groupcomm_profile
  – exp
  – **app_groups**  // Names of application groups
  – joining_uri
  – ? group_policies
  – ? max_stale_sets
  – ? as_uri  // Link to the AS

- When using PATCH, easy “replacement” update for most parameters
  - Specify the pair (“label”, new_value), like when creating the group
  - ‘app_groups’ is a list of names and requires special handling
Configuration update with PATCH

› Two ways to update ‘app_groups’
  – List of associated applications groups

› **Overwrite** – New array of names as hard replacement
  – app_groups : [“room1”, “room8”]  *Custom CBOR*
  
  – app_group “room1”
  – app_group “room8”  *CoRAL*

› **Addition/deletion** – [ [*name_to_remove], [*name_to_add] ]
  – app_groups_diff : [ [“room1”], [“room5”] ]  *Custom CBOR*
  
  – app_group_del “room1”
  – app_group_add “room8”  *CoRAL*

› Overwrite and addition/deletion **not together** in the same PATCH payload

Current value [“room1”, “room2”]

The result is [“room1”, “room8”]

The result is [“room8”, “room5”]
Configuration update with PATCH

› 4.00 (Bad request)
  – Any malformed or invalid payload
  – iPATCH is used as request method, but:
    › ‘app_groups_diff’ is included (Custom CBOR)
    › ‘app_group_del’ and/or ‘app_group_add’ are included (CoRAL)

› 4.09 (Conflict)
  – New parameter values would yield an inconsistent group configuration

› 4.22 (Unprocessable entity) might be returned just as per RFC 8132
  – The server is unable to or is incapable of processing the request