GNAP model & trust relationships Privacy considerations & Security considerations

Denis PINKAS

President of DP Security Consulting SAS. France

IETF 110 GNAP WG Virtual meeting March, the 9 th , 2021

The three main components of the model

- **Client** : application operated by an end-user that consumes resources from one or several RSs, possibly requiring access privileges from one or several ASs.
- **Authorization Server (AS):** server that grants delegated privileges to a particular instance of client software in the form of an access token and other information (…).
- **Resource Server (RS) :**

  [*current definition in the draft*] **:** server that provides operations on ***protected*** **resources**, where operations require a **valid** access token issued by an AS.

  [ISO definition from 24619:2011, 3.3.6], computer that ultimately provides access to the object referenced by a specific client application request.

  [Proposed definition]: server that provides operations to objects referenced by specific client application requests.

One additional concept: the concept of a SERVICE

- An API is an abstraction layer for accessing a service *which may be fulfilled by a set of RSs*.

Scalability and trust relationships

In OAuth:

- The token format is defined between a RS and an AS using out of bands means,
  hence a prior relationship must exist between every RS and every AS. This is not scalable.

In GNAP:

- The access token format should be advertised using some protocol by both the RS and the AS
  (e.g. using a Discovery mechanism) so that a common format may be discovered and/or negotiated.

- In terms **of trust relationships**, the RS trusts a set of ASs ***for some sets of privileges***
  (i.e. rights or/and attributes) contained into an access token.

- In general, an AS does not need to have a prior knowledge of the RSs, nor any kind of trust relationship
  with the RSs.

- However, in the case where an AS delivers capabilities on some objects on behalf of a RO,
  that AS needs to have a trust relationship with that RO (but not that RS).

The two faces of the Resource Owner (RO) (1/2)

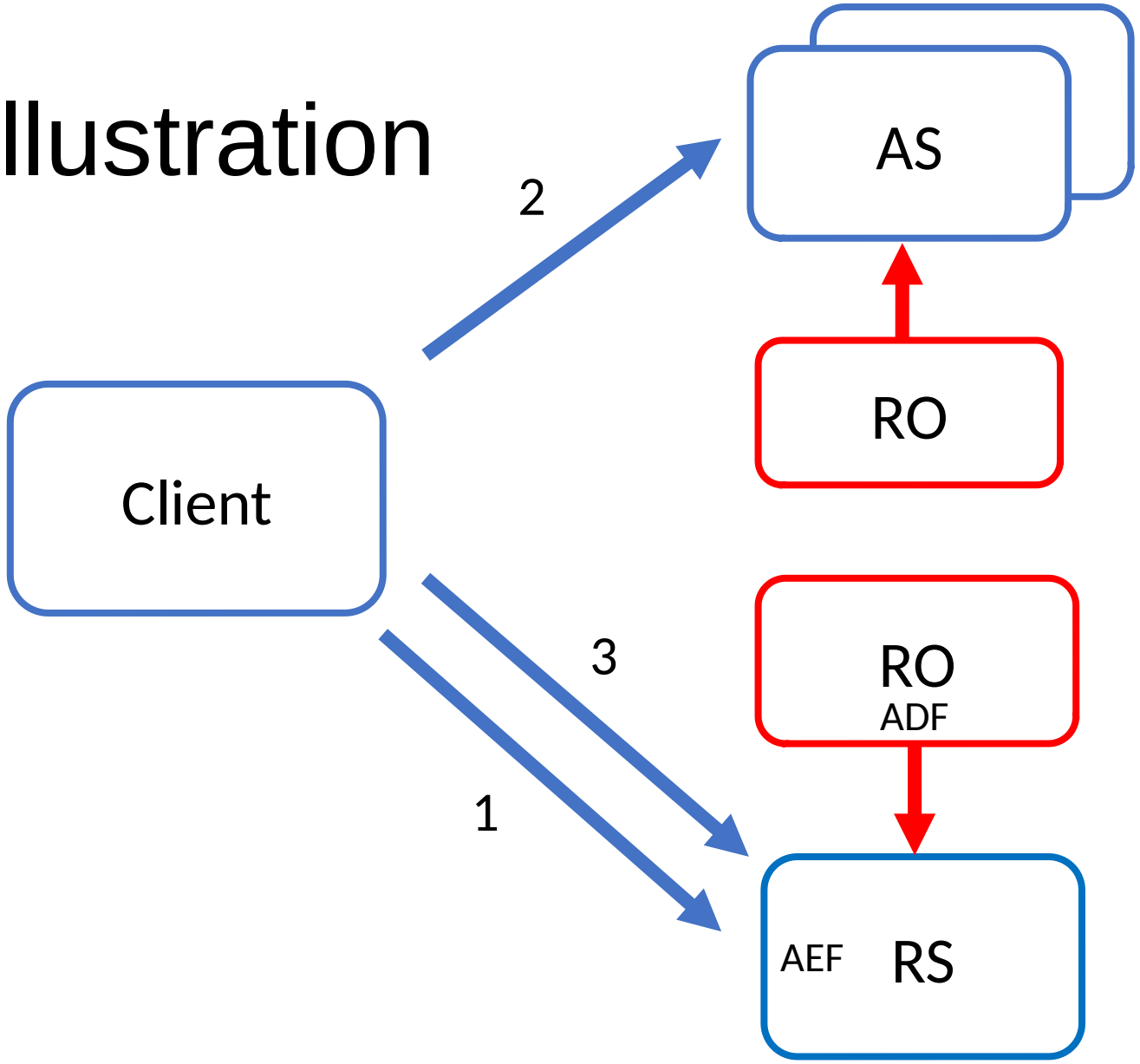
- **Resource Owner (RO) :** "*subject* entity that may grant or deny operations on resources it has authority upon".

- The RS supports an Access Enforcement Function (AEF) which follows access control rules established by an Access Decision Function (ADF) administered by a **Resource Owner (RO).**
- These access control rules may be either ACL based and/or capability based.
  The RO administering the RS may be a process, a human being or a device (IoT case).
- For each method requested by a client on an object, the rules enacted by the RO indicate which methods may be granted. Some rules may allow the use of some methods without the need to present an access token (public access or access protected by an authentication mechanism), while some other rules may require one or more access tokens.
- In this last case, these rules are a combination of AS URLs (using OR or/and AND operators), and when these rules are:
  - **ACL based** : they indicate which **attribute types and attributes values** issued by that AS must be presented to use a method on the object, and/or
  - **Capability based** : they indicate which **methods** issued by that AS must be presented to use a method to the object.

The two faces of the Resource Owner (RO) (2/2)

- When the access control rules are **Capability based**, the AS must cooperate with a RO able to deliver capabilities on objects of the RS under its control.
- In that case, the RO may either be a process or a human being.
- IMPORTANT: When a RO associated with an AS delivers a capability on a resource under the control of a RS, that capability is analysed and filtered by the ADF of the RS (i.e. a RO) that indicates which method(s) delivered by a given AS can be accepted.

Illustration



Optional RO that MAY issue capabilities for one RS

The RO may either be a process or a human being

RO acting as an ADF for a RS when both ACLs and/or capabilities are being used.

The RO may be either a process, a human being or a device (IoT case)

Resource Owner (RO) : *subject* entity that may grant or deny operations on resources it has authority upon.

Privacy considerations

- *Privacy by Design* (PbD) means that the privacy requirements are identified once the model components, including their trust relationships, have been identified and defined but *before* defining the protocols.
- Section 14 (Privacy considerations) states : "*There are a lot of privacy considerations to add*".
- "*Privacy after design*" is usually unable to address privacy considerations which have not been considered before the design of the protocols.
- Two important privacy considerations are about:
  - User choice and consent, and
  - User notice.

User choice and consent, and user notice

- In the current draft, a user consent phase is only considered to take place at the (single) AS.
- However, as soon as a RS is trusting more than one AS, the end-user must be able to make at least an AS(s) *choice before contacting any AS*.
- Furthermore, the end-user must be able to be informed of the reasons or the rational for each privilege before requesting an access token that should contain the selected privileges: this step is called user notice:

  Every end-user should be able to agree about the rights and/or the **attributes types** *and values* he accepts to be placed into an access token.

- *These two steps are currently unsupported/missing in the draft.*

The various types of end-user identifiers that need to be distinguished

An end-user identifier present in an access token may either be:

(1) a globally unique identifier (e.g. an email address, a social security number),
(2) an identifier locally unique to that AS for all the RSs,
(3) an identifier unique for every AS - RS pair, or
(4) a temporary identifier for a single access (i.e. a large random identifier).

Let us explore the consequences, when considering:

One **privacy** property:

Unlinkeability between RS user accounts : Two RSs should not be able to link their user accounts, by using the content of the access tokens they receive from the same AS or from different ASs.

and one **security** property:

Token forwarding between collaborating clients: two collaborating clients should not be able to *anonymously* use access tokens of the other client on the same RS.

Unlikability between RS user accounts versus Client collaboration attacks

		Privacy property	Security property
	Type of end-user identifier contained into an access token	**Unlinkability between RS user accounts**	**Token forwarding between collaborating clients defeated**
(1)	Globally unique identifier (e.g an email address)	✗	✓
(2)	Locally unique identifier used by that AS for all the RSs	✗	✓
(3)	Unique end-user identifier for every AS - RS pair	✓	✓
(4)	Temporary end-user identifier for a single access	✓	✗
(5)	No end-user identifier	✓	✗

Consequence: if an access token *only* contains *one or more* capabilities, client collaboration attacks will succeed. In order to defeat client collaboration attacks, the access token must *also* contain a type (1), (2) or (3) end-user identifier.

If the choice is type (1) or (2), then RS user account unlinkability characteristics cannot be obtained.
If the choice is type (3), then the identifier of the RS cannot be hidden to the AS.

Note: it is possible to *have the cake and eat it too*, if specific secure elements are being used by the end-users.

Some differences whether attributes or rights are being used

Privacy characteristics which depend whether *only* ACLs or capabilities are being used:

Privacy characteristics	When using ACLs only	When using capabilities only
Unlinkeability between RS user accounts	✓ with type (3) only	✓
AS - RS untraceability	✓ except for type (3)	✗
AS - Client Operation (method) hidding	✓	✗

Privacy characteristics which do not depend whether ACLs or capabilities are being used:

Privacy characteristics	... which may be obtained using
AS - Client Access time invisibility	✓ No Token introspection operation
End-user confidence in the system (i.e. Transparency)	✓ No Token introspection operation & Token transparency for clients

End-user identifiers types

CONSEQUENCES:

- A RS should be able to indicate to a client which *end-user identifier types* may be presented to be able to perform an operation,
- The client should be able to indicate these *end-user identifier types* to the selected AS, and
- The RS must be able to distinguish between these *end-user identifier types* obtained from an AS.

The *Security Events WG* is currently developing a draft (draft-ietf-secevent-subject-identifiers-06) about “Subject Identifiers for Security Event Tokens” which defines a "sub_id" claim.

A liaison with the *Security Events WG* should be established, to explore how the various end-user identifier types previously identified would be supported.