IETF 112 GNAP Session November 11, 2021

Concerns with draft-08 and how to address them

Denis PINKAS
President of DP Security Consulting SAS. France

Attributes are within the scope of the core document :

Privilege: right **or attribute** associated with a subject

Access Token: a data artifact representing a set of rights **and/or attribute**

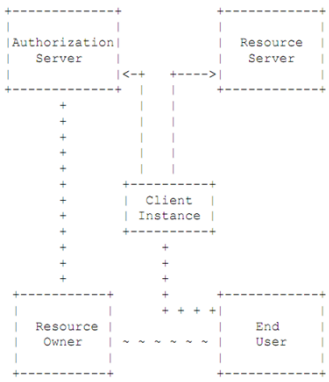
Currently draft-08 only supports rights (i.e. capabilities) i.e. CBAC (Capability Based Access Control) **but not attributes** with its two flavours:

- ABAC (Attribute Based Access Control) and
- RBAC (Role Based Access Control).

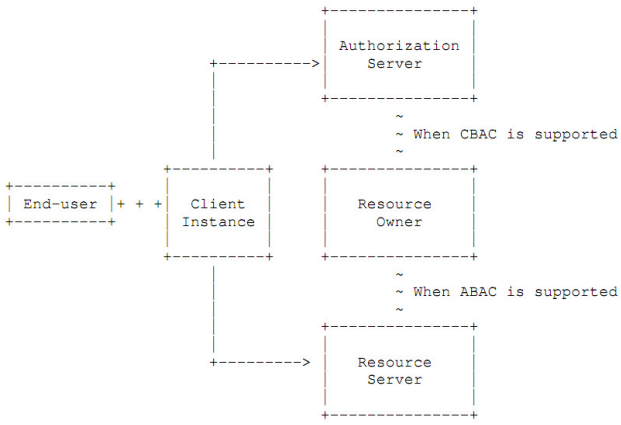
- The support of attributes should be added to the core document.

2

This leads to the following model



Draft-08 model



New model

3

Attributes are not currently supported in the core-document but should be supported

- The core-document should be ready to support the "Proposal for a Regulation amending Regulation (EU) No 910/2014 as regards establishing a framework for a European Digital Identity" issued on 3.6.2021. This proposal considers a new qualified trust service for the provision of electronic attestations of attributes.

  Note: This implies the ability to receive end-user attributes from multiple sources.

- The core protocol should define two fields in an access token request to distinguish between rights (i.e. capabilities) and attributes.
- A set of common attribute **types** should be defined:

  e.g. first name, family name, birth date, birth location, home address, email address, social security number, citizenship(s), etc …
- End-users should be able to query an AS to know which attributes types and *values* are known by a given AS (Requesting end-user Information, *i.e. not RO Information*).

4

What should be done to support attributes ?

(1) To allow a RS to indicate which attributes types (*and optionally values*) should be included into the access token in order to allow a given operation on a given protected resource (RS), e.g. by using an **HTTP OPTIONS request.**

(2) To allow end-users to know which AS(s) is/are appropriate for each attribute type.

(3) To allow end-users to know the reason why these attributes types should be disclosed using their **preferred language** (**User Notice**).

(4) To allow end-users to accept (or deny) fetching these attributes types using their **preferred language** (**User Choice and Consent**).

(5) To allow clients to request to one or more ASs, such attribute types.

(6) To allow clients to **hide to the AS the identity of the RS** (using an unsigned part of the Access Token).

5

Benefits of the support of attributes

- A RS trusts a set of ASs *for some types of attributes*.
  No *bilateral* pre-relationship between an AS and a RS is necessary.
- A RO working in collaboration with a **RS** is needed to support attributes
  (*while a RO working in collaboration with an **AS** is needed to support rights*),
- The AS can be kept ignorant of which actions will be performed by a client and on which resources.
- In some cases (*i.e. not all*), the AS can be kept ignorant of which RS will be accessed (the true identifier of the RS can be concealed by the client to the AS).
- The following privacy properties can be supported:
  User Notice, **User Choice and Consent**.

6

draft-08 makes a general assumption that an end-user and a RO are the same entity

- Extract from page 13 of draft-08:

  Note that the RO and end-user are often the same entity in practice, but GNAP makes no general assumption that they are.

- Unfortunately, this Note is wrong.

- Both sections 1.5.1 and 1.5.2 are about cases where the user is both the end-user and the RO.

- Section 1.5 is failing to provide a single example where the end-user is not also the RO and where the RO is an automated process.

7

Trust relationships in draft-08

Section 1.4 of draft-08 is supposed to be about "*trust relationships*", but it is not.

It is based on "Promise Theory" document (302 pages) where " *A promise is a stated intention* ".
" *Each agent defines its own valuation function of promises given or received* ".

On page 108 (!), there is a first attempt to define what Trust means:

> *Proposal* 1 (Trust). An agent's expectation that a promise will be kept.
> It may be assigned a value lying between 0 and 1, in the manner of a Bayesian probability.

In the IT community, trust is a binary condition (0 or 1): either you trust something or you don't.
A trust relationship is simple and composed of three parts :

An entity **A** … is trusting an entity **B** … for some kind of behavior **C**.

Section 1.4 of draft-08 is failing to indicate the trust relationships that exist between:

- client instances and ASs,
- end-users and ASs.

8

In draft-08, the link, if any, between client instance authentication and end-user authentication is undefined

- How may a client instance be associated with an end-user ? This is left undefined.
- How may an AS be confident that a legitimate user is using a given client instance ? This is left undefined.
- Extract from the Introduction on page 5 of draft-08:

  This protocol allows a piece of software, the client instance, to request delegated authorization to resource servers and to request direct information. (…)

  The end-user operating the software may interact with the authorization server to authenticate, provide consent, and authorize the request.

Hence, the authentication of end-users by an AS is not required.

- Draft-08 will be unable to comply with the EU Payments Services Directive (PSD2) which requires MFA (Multi-factor authentication).

9

Why the draft-08 model is insecure

From section 12.3 (Protection of Client Instance Key Material):

" Client instances are identified by their unique keys, and anyone with access to a client instance's key material will be able to impersonate that client instance to all parties".

Despite the title of that section, the protection of the Client Instance key material is not addressed in section 12.3. Anyway, such protection cannot be guaranteed.

Even if a hardware security module (HSM) is being used to protect a client private key, this does not prevent the *usage of that private key*.

As soon as two client instances collaborate, the security foundations of draft-08 collapse.

A solution resistant to client collaborative attacks is needed ... and is possible.

10

However such solution is negated by draft-08

From section 13.4.2. Correlation by Resource Servers (New text) :

"However, note that the lack of inclusion of a user identifier in an access token may be a risk if there is a concern that two users may voluntarily share access tokens between them in order to access protected resources ".

On the contrary, the inclusion of a user identifier in an access token is able to prevent two end-users or two clients to voluntarily share access tokens between them.

Section 13.4.2 terminates with:

"(Note that the binding of an access token to a non-extractable client instance key also prevents the access token from being voluntarily shared) "

This is incorrect. On the contrary, the addition of a binding user identifier ("buid") field into an access token is able to prevent, *in some situations,* the access token from being voluntarily shared.

11

Two new concepts need first to be considered: long-term and short-term RS user accounts

Note: This topic has been discussed on github. See :
https://github.com/ietf-wg-gnap/gnap-core-protocol/issues/322

Details are in: https://www.ietf.org/id/draft-pinkas-gnap-core-protocol-00.txt#:~:text=1.7

Long-term user account

The end-user is willing to retrieve, deposit or modify some data that will be saved by the RS.
A typical example is an access by an end-user to his bank account.

Short-term user account

The end-user is willing to perform a transaction with the RS. Once the session will be closed, the RS will not maintain any information about that temporary user account.
A typical example is a train reservation without the creation of a user account.

Access tokens can be bound to a RS user account

12

The "buid" field : Binding User IDentifier

- A "buid" field (Binding User IDentifier) is composed of two parts:
  - a type (among 5 choices) chosen either by the client or by the end-user,
  - a value always generated by the AS.
- It allows a RS to verify that the access token is associated with the right (short-term or long-term) user account.
- The use of a "buid" field (Binding User IDentifier) is able to counter:
  - all client collaborative attacks on long terms user accounts, and
  - some client collaborative attacks on short-term user accounts.

More details are in https://www.ietf.org/id/draft-pinkas-gnap-core-protocol-00.txt

13

The five types of "buid"

The four types used in the context of long-term RS user accounts are :

(1) a unique user identifier for each User/ RS pair,

Note: this option cannot be implemented in the context of a "software-only" solution. It requires the use of a secure element with specific security properties.

(2) a unique user identifier for each AS / RS pair,

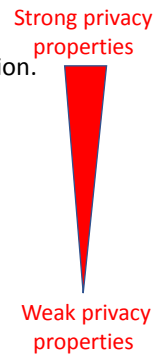
Note: the client MUST disclose to the AS the identity of the RS.

(3) a locally unique user identifier for whatever RS,

Note: the client MAY hide to the AS the identity of the RS.

(4) a globally unique user identifier (e.g. an email address).

Note: the client MAY hide to the AS the identity of the RS.



The fifth type used in the context of short-term RS user accounts is :

(5) a session user unique identifier.

Note: the client MAY hide to the AS the identity of the RS.

14

How the "buid" detection mechanism works

- Any RS user account SHALL only be established using an access token issued by an AS that contains a "buid" field.
- Then after, the RS SHALL verify that any subsequent access token coming from the same AS contains the same "buid" field, otherwise it SHALL reject it.
- These rules are valid both for long-term and short-term RS user accounts.

Consequence: If a user A transmits to a user B an access token that he legitimately got, that user B will be **unable to use** that forwarded access token on his own RS user account.

In the context of a short-term RS user account :

- A single access token can be successfully used by anybody on a given RS, if it does not contain a set of attributes that allows to sufficiently identify the end-user (e.g. only the attribute "over 18") … unless it can be associated with a previous access token that contained the same "buid" and which allowed to sufficiently identify the end-user.

Note: This mechanism does not protect against impersonation.
The use of ephemeral key pairs allows to protect against impersonation.

15

The EU legislation should be taken into consideration

- GNAP should be usable within the EU, hence :
  - The General Data Protection Regulation (GDPR) i.e. REGULATION (EU) 2016/679 of 27 April 2016 on the protection of natural persons with regard to the processing of personal data should not be ignored.
  - The EU Payments Services Directive (PSD2) should not be ignored.
  - The *Proposal* for a Regulation amending Regulation (EU) No 910/2014 as regards establishing a framework for a European Digital Identity of 3 June 2021 should not be ignored.

16

Towards a modified model

1. Both attributes (ABAC) and capabilities (CBAC) SHOULD be supported

2. End-users SHALL authenticate to ASs

Section 1.6 from draft-pinkas-gnap-core-protocol-00 defines pre-arrangements before the protocols can be used. They are supported using "out-of-bands means" which are outside the scope of the protocol.

- Every end-user MUST have an account opened with at least one AS.
- When the account is settled between the end-user and the AS, a user identifier and an authentication method SHALL be agreed. The end-user MUST receive information that allows him to perform a first authentication exchange with the AS with success.

3. Trust relationships need to be clear and understandable. A RS SHALL be able to trust more that one AS

For a definition of the trust relationships that are common to both CBAC and ABAC and those which are specific either to CBAC or ABAC, please refer to section 1.5 from draft-pinkas-gnap-core-protocol-00.

4. In order to defeat collaborative attacks, access tokens SHOULD be protected using the "buid" mechanism

5. In order to protect against impersonation, access tokens SHOULD be protected using mutual TLS with ephemeral client key pairs

17