ACE comparison

Bert Greevenbosch, bert.greevenbosch@huawei.com

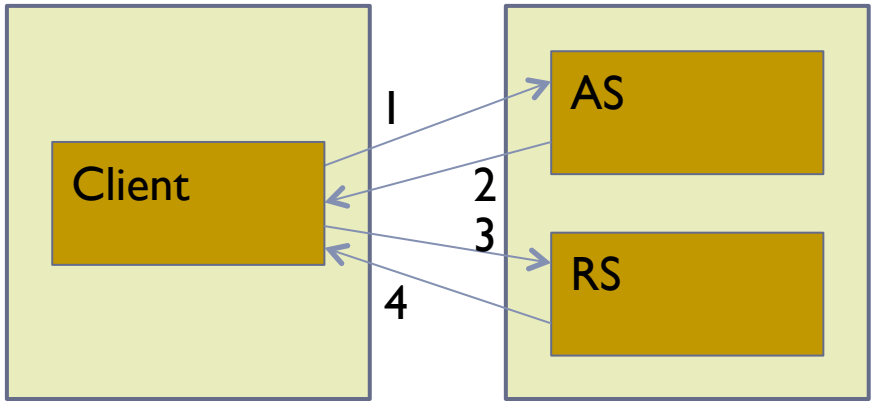
Introduction

- draft-greevenbosch-ace-comparison compares the currently proposed solutions for ACE.
- It compares:
  - DCAF
    - draft-gerdes-ace-dcaf-authorize, draft-gerdes-ace-actors
  - OAuth
    - draft-tschofenig-ace-oauth-bt, draft-tschofening-ace-oauth-iot, draft-wahlstroem-ace-oauth-iot
  - EAP
    - draft-marin-ace-wg-coap-eap
  - 2-way authentication for the Internet of Things (TWAI)
    - draft-schmitt-ace-twowayauth-for-iot
  - Pull Model
    - draft-greevenbosch-ace-pull-model
- https://datatracker.ietf.org/doc/draft-greevenbosch-ace-comparison/

Architectural models

- Currently, there are three architectural models on the table:
  - Push
  - Indirect push
  - Pull

Push model



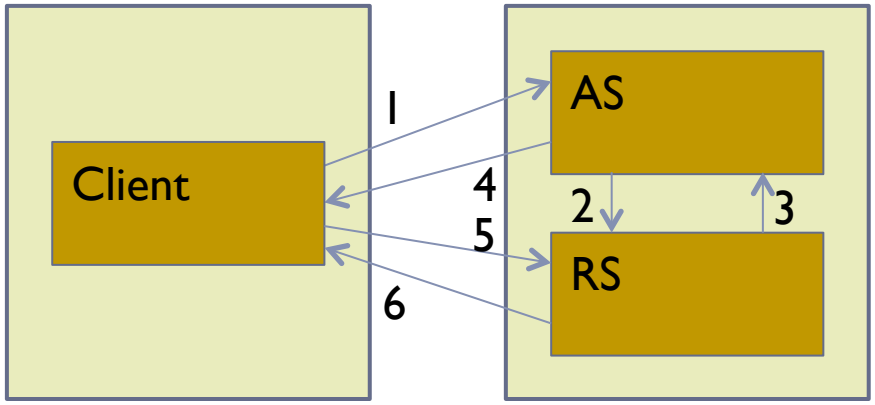
Advantages:

- Message transmission and processing workload for Resource Server is low but message transmission and processing workload for client is higher.
- Since Resource Server is usually the most constrained, this model is suitable for constrained environment.

Disadvantages:

- Client may not have direct connection with Authorization Server.
- Ticket revocation mechanism may be needed, e.g. in case of security breach or policy modification.
- Without such mechanism, client is still able to access resource in accordance to old policy.

Indirect push model



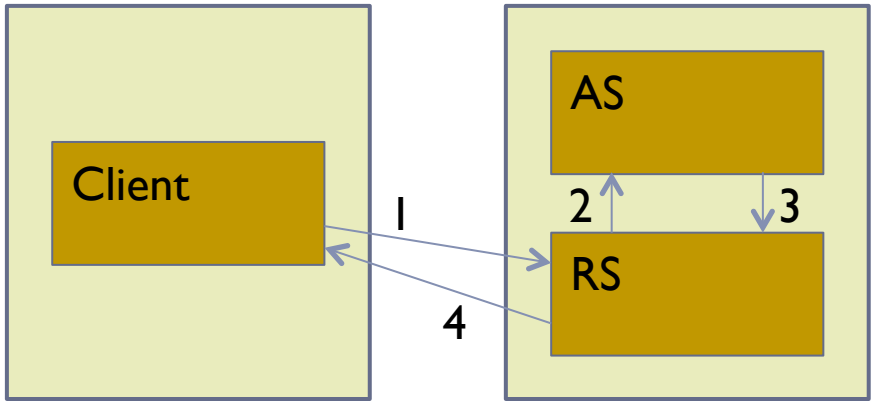
Advantages:

- Does not require client to receive Access Ticket from Authorization Server and send it to Resource Server.
- Efficient if Access Ticket is large and the client is resource constrained.

Disadvantages:

- More processes in the whole authorization flow.
- Adds a burden to Resource Server to cache Access Ticket.

Pull model



Advantages:

- Still works when client cannot have direct connection with Resource Server.

Disadvantages:

- Authorization Server works as agent, and is involved in resource access process.
- Logically, Authorization Server should just take care of authentication/authorization, and should not mix other functionalities.

Models in the proposals

- DCAF: pull model
- OAuth: pull model or indirect push model
- EAP: pull model (AAA acts as AS)
- TWAI: indirect push
- PULL: push model

Number of parties

- ACE: 3 or 4 (AM may be combined with client)
- OAuth: 3
- EAP: 2 or 3 (depending on usage of AAA server)
- TWAI: 4/5 (gateway and access control server could be combined)
- PULL: 3

Ticket format

- DCAF: defined for CoAP
- OAuth: needs definition
- EAP: n.a.
- TWAI: needs definition
- PULL: same as ACE

Protocol for exchange of authorisation information:

- DCAF: between C and AM, and between C and RS: DTLS; between AM and AS: proprietary.
- OAuth: between C and AS: CoAP+DTLS. Between C and RS: DTLS?
- EAP: CoAP
- TWAI: to be defined
- PULL: CoAP + DTLS

Client credentials

- DCAF: through AM (possibly AM's X.509 certificate)
- OAuth: out of scope
- EAP: out of scope
- TWAI: X.509 certificates
- PULL: out of scope

Definition of new CoAP options

- DCAF: no
- OAuth: yes, "Bearer" and "Error"
- EAP: maybe, "AUTH"
- TWAI: no
- PULL: re-uses new "Node-Id" option

Further considerations

- DCAF, TWAI and PULL were originally designed with Internet of Things applications in mind.
- EAP and OAuth are trying to tweak a solution that was not designed for IoT towards IoT.

Discussion material

- Which model do we like?
- How many entities do we like?
- What kind of access permission granuality do we want?
- How do we want to move forward?

Thank you!