AAI and Token Authorization

Oliver Freyermuth, Peter Wienemann

University of Bonn
{o.freyermuth,peter.wienemann}@uni-bonn.de

22nd June, 2022

Physikalisches Institut,
UNIVERSITÄT BONN

Identity & Access Management Systems (IAMs)

- WLCG (+Belle) use Indigo IAM (INFN maintained, developed in INDIGO-Datacloud Horizon 2020 / EOSC)
- PUNCH (Helmholtz), EUDAT/B2ACCESS use Unity IAM

Common functionality

- Both IAMs support OpenIDConnect workflows
- JSON Web Tokens (JWT) can be fetched by users and used as access tokens

However, there are slight differences in the token headers and payload.

Unity IAM: Access Tokens (Header & Payload)

Decode token payload:

```
for T in $(oidc-token punch | tr '.' '\n' | head -2); do
  echo $T | base64 -d | jq;
done
```

Unity IAM: Access Tokens (Header & Payload)

```
{ "typ": "at+jwt", "alg": "RS256" }
{
  "sub": <uuid>,
  "aud": "public-oidc-agent",
  "scope": "openid offline_access email profile eduperson_scoped_affiliation
↪   eduperson_entitlement eduperson_unique_id",
  "iss": "https://login.helmholtz.de/oauth2",
  "exp": 1644413969,
  "iat": 1644409969,
  "jti": <uuid>,
  "client_id": "public-oidc-agent"
}
```

- **aud**ience claim does not seem to be a resource indicator (validation?)
- No Authorization Claims (more later)

Indigo IAM: WLCG Access Tokens (Header & Payload)

Decode token payload:

```
for T in $(oidc-token wlcg | tr '.' '\n' | head -2); do
  echo $T | base64 -d | jq;
done
```

Indigo IAM: WLCG Access Tokens (Header & Payload)

```
{ "kid": "rsa1", "alg": "RS256" }
{
  "wlcg.ver": "1.0",
  "sub": <uuid>,
  "aud": "https://wlcg.cern.ch/jwt/v1/any",
  "nbf": 1644409963,
  "scope": "openid offline_access profile eduperson_scoped_affiliation
↪   eduperson_entitlement email wlcg wlcg.groups",
  "iss": "https://wlcg.cloud.cnaf.infn.it/",
  "exp": 1644413563,
  "iat": 1644409963,
  "jti": <uuid>,
  "client_id": <uuid>,
  "wlcg.groups": [
    "/wlcg"
  ]
}
```

Main difference: Authorization Claims

WLCG tokens embed authorization claims in the access tokens

- Decentralized usage model (scaling!):
  No user info queries by resource providers to central instance needed.
- Group membership and granular authorizations e.g. for storage access possible, inspired by SciTokens Claims and Scopes Language
- 2019: Full WLCG JWT profile described (see Zenodo)
- 2021: Concept reflected in RFC9068, section 2.2.3.1
- Future plan: Revise WLCG JWT profile (see TWiki)
  - Evolve closer to RFC9068
  - Drop any WLCG-specific naming
  - Add granular compute scopes

$\Rightarrow$ Without this, all resource providers need to query central `userinfo` endpoint.

Unity IAM: `userinfo` contains groups (entitlements)

Contact `userinfo` endpoint (with access token):

```
curl -H "Authorization: Bearer `oidc-token punch`" -X GET
↪  https://login.helmholtz.de/oauth2/userinfo | jq .
```

Unity IAM: `userinfo` contains groups (entitlements)

```
{
  "sub": <uuid>,
  "email_verified": true,
  "name": "Oliver Freyermuth",
  "eduperson_scoped_affiliation": "member@uni-bonn.de",
  "eduperson_unique_id": "XYZ@login.helmholtz-data-federation.de",
  "preferred_username": "o.freyermuth",
  "given_name": "Oliver",
  "family_name": "Freyermuth",
  "email": "o.freyermuth@uni-bonn.de",
  "eduperson_entitlement": [
    "urn:geant:h-df.de:group:HDF#login.helmholtz.de",
    "urn:geant:dfn.de:nfdi.de:punch:group:PUNCH4NFDI#login.helmholtz.de",
    "urn:mace:dir:entitlement:common-lib-terms"
  ]
}
```

Usage Workflows

- Users in PUNCH may have multiple roles in one group, e.g.
  - Inject raw data into Storage4PUNCH
  - Write preprocessed data for a collaboration
  - Read access to preliminary data / public data only
  - Database write access, metadata catalogue write access, ...

  $\Rightarrow$ Need to select role to assume / delegate.
- Parts of the PUNCH infrastructure may require reduced permissions only (delegation of less powerful tokens for safety)

  Example: A batch compute job may not be allowed to cancel other compute jobs, an interactive job may be allowed
- Hundreds of thousands of analysis tasks may want to be authorized in parallel.
- Delegation may implicitly be done by the user (e.g. via portal, submit node,...). Concept known from macaroons.

Delegation, Roles and dynamic Groups

- 'VOMS' concept: Groups & Roles, e.g.: Group $\hat{=}$ Experiment, Role $\hat{=}$ prod. user
- Translation into **default groups** and **optional groups** in tokens
- Scope added to allow group selection:

```
wlcg.groups[:<group_name>]?
```

- Scopes to allow capability selection:

```
storage.read:/punch/somewhere compute.create:/
```

- Future-proofed by versioning tokens (special scope for that)

Standardized components

- `groups`, `roles`, `entitlements` claims in RFC9068, section 2.2.3.1
- Using standardized `acr` claim for 'assurance' (single factor, multi factor,...)
- Using standardized `aud` claim for targeted audience (resource provider)

Features missing in the Unity IAM

- **aud** claim does not seem to be a resource indicator, but matches client id (validation?)
- **acr** claim not implemented yet
- For decentralized authorization, Authorization Claims need to be inside tokens, e.g. as standardized in RFC9068, section 2.2.3.1
- Granular authorization and delegation in tokens required for workflows
- In turn requires authorization information in token controllable via requested scopes
- Software currently in use in communities expects this
- Usage of more granular scopes / capabilities may be wanted in the future

Thank you

for your attention!