Token based solutions for OIDC

Diana Gudu, Marcus Hardt, Gabriel Zachmann

Mar 2023

Use Cases

Use Case: API Access

- Why would you want to do it?
  - Web apps: Web frontend + REST access to backend
  - Cloud services:
    - Create Openstack VM from commandline
    - Access storage
    - Github
    - ...
- **Protect** your REST API with OIDC: => **`flaat`**
- **Access** a REST API with OIDC: => **`oidc-agent`** + `curl`

Use Case: Grid

- X.509 => Tokens
- All use cases from grid should work with tokens:
- Service migration -> Service developers => **`flaat`**
- How does the user get tokens => **`oidc-agent`**
- But: can we just replace X.509 certificates with tokens?
  - Tokens cannot be revoked
  - Tokens (should) live much shorter
  - Long-running jobs need fresh tokens
    - => Look at **`mytoken`**

Use Case: Shell Access

- I started my Openstack VM, now what?
  - I need to access remote services (e.g. using tokens) => access tokens
  - I need an access token throughout the lifetime of the VM => **`mytoken`**, maybe **`vault`**
- Similar for HPC
  - I need to ssh into my HPC-cluster / VM => **`ssh-oidc/motley-cue`**

Fun with OIDC

VIA 9GAG.COM

Getting access tokens

https://github.com/indigo-dc/oidc-agent

- Problem 1: OIDC is not designed to give access tokens to users
- Problem 2: Access tokens expire rather soon (actually this is a good thing!)

- Solution: **`oidc-agent`** (think "`ssh-agent`")
- Tool for OIDC access tokens on the commandline
  - WLCG, Unicore, FedCloud, ARC-CE, Japan HPC, Fenix, ... (we lost track, ...)
  - Obtain **refresh token**
    - Run an OIDC web flow
    - Crypt it
    - Store it
    - Load to RAM when needed (also crypted when in RAM)

- Key Features
  - Packaged for Major Systems [**Debian**, **Ubuntu**, **Fedora**, **Centos**, **Suse**, **MacOS**, **Windows**]
  - Secret storage
  - Graphical Query
  - mytoken integration

```
# Example:
# Step 1: Obtain a refresh token from your issuer:
    oidc-gen --pub --iss https://wlcg.cloud.cnaf.infn.it  --scope "eduperson_entitlement email" wlcg-demo
        <follow the flow in your browser ...>

# Step 2: Get access tokens:
    oidc-token wlcg-demo

# Step 3: Take a look at the token, and the userinfo endpoint:
for T in $(oidc-token wlcg-demo| tr '.' '\n' ); do
    echo $T | base64 -di 2>/dev/null | jq --indent 4 2>/dev/null
done
```

```
# Step 3: Or just do it in python
    pip install flaat
    flaat-userinfo `oidc-token wlcg-dem
```

Resul t

Information stored inside the access token:


```
{
    "body": {
        "aud": "https://wlcg.cern.ch/jwt/v1/any",
        "client_id": "c44fc787-b3f8-483d-a78b-22c29fd4e524",
        "exp": 1679973409,
        "iat": 1679969809,
        "iss": "https://wlcg.cloud.cnaf.infn.it/",
        "jti": "36932e6e-44d9-4688-b867-f7a4fd76f2ad",
        "nbf": 1679969809,
        "scope": "openid offline_access profile eduperson_scoped_affiliation eduperson_entitlement email wlcg wlcg.groups",
        "sub": "61a5aa12-27c8-41c1-b05b-9eb6f724d29f",
        "wlcg.groups": [ "/wlcg" ],
        "wlcg.ver": "1.0"
    },
    "header": { "alg": "RS256", "kid": "rsa1" },
    "signature": "Jysd5TXn0iTbaPkjXKjjnVdM9ae5y8J4LK_jdUX-m5JmvE1d_njzq151tE629lmYjKqC0AOF88dShG2efJ7d1saFBIQ3sV50otvbSSDC81BAFLqznecS_FmbpI
    "verification": {
        "algorithm": "RS256"
    }
}
```


Information retrieved from userinfo endpoint:


```
{
    "email": "marcus.hardt@kit.edu",
    "email_verified": true,
    "family_name": "Hardt",
    "given_name": "Marcus",
    "iss": "https://wlcg.cloud.cnaf.infn.it/",
    "name": "Marcus Hardt",
    "preferred_username": "marcus2",
    "sub": "61a5aa12-27c8-41c1-b05b-9eb6f724d29f",
    "updated_at": 1595850339,
    "wlcg.groups": [
        "/wlcg"
    ]
}
```

More fun

flaat

https://flaat.readthedocs.io

- Authorisation for REST APIs
- Flexible python framework
  - Supports **flask**, **AIO**, **FastAPI**

```
xample Code (skipping boilerplate)
utes.get("/authorized_vo")
aat.requires(
 get_vo_requirement(
     [
         "/wlcg",
         "/cms",
     ],
     "wlcg.groups",
     match 1,
 )

ac def authorized_vo(request):
 return web.Response(text "This worked: use
```

```
# Example call and response
marcus@nemo:~$ http http://localhost:8080/authorized_vo  "Authorization: Bearer
HTTP/1.1 200 OK
Content-Length: 46
Content-Type: text/plain; charset utf-8
Date: Tue, 28 Mar 2023 02:40:39 GMT
Server: Python/3.11 aiohttp/3.8.3

This worked: user has the required entitlement
```

What else?

mytoken [1/2]

https://mytoken.data.kit.edu

- What if you have a long-running job
  - … that also spends some time in a queue
- **`mytoken`** (think "`myProxy`" done right)
- Solves this!
- `mytoken-server`:
  - gets the `refresh token` (typical web-flow)
  - encrypts + stores `refresh token` with `mytoken`
  - returns the `mytoken`-token to user
- `user (client)`:
  - use the `mytoken` with `mytoken-server` to get **`access token`**

mytoken [2/2]

- Why is it better than sending **`refresh-tokens`** with the job?
- Features to carefully balance security requirements with use case:
- **Capabilities**
  - **WHAT** can the mytoken do (get ATs, create new MTs, history, introspection, settings, ...)
- **Restrictions**
  - **LIMITATIONS** on the token are based on
    - time, IP address space, geolocation, number of usages, scopes, **audiences**
- More info: https://mytoken-docs.data.kit.edu
- Example: https://mytoken.data.kit.edu
- Using **`mytoken`** client from the cmdline

```
$ mytoken AT --MT $MYTOKEN
eyJ.............
```

Combining these solutions….

…for SSH with federated identities

SSH

Kind of holy grail, because:

- Use federated identity
  - SSH-Server has no direct relation with Organisation where user comes from
- To log into a Unix account
  - How to find the correct unix account?
- Authorised by
  - Virtual Organisation membership (**`entitlement`**)
  - Assurance
  - Individual user (`sub + iss`)
- How to revoke access?
  - User gone
  - Security incident

Approaches

`flaat` + `pam-module`

- PAM Module **`pam-ssh-oidc`** developed by PSNC (in Pracelab.PL)
- `pam-ssh-oidc` enables two things:
  1. Prompt for “Access Token”
  2. Put access-token into ssh password field
- `pam-module` uses **`flaat`** for authorisation
- Prerequisite: the remote user has to exist

```
# Example
$ oidc-token google # or mytoken AT --MT $MYTOKEN
$ ssh cool001@ssh-oidc-demo.data.kit.edu

(cool001me@ssh-oidc-demo.data.kit.edu) Access Token:
(cool001me@ssh-oidc-demo.data.kit.edu) Password:
(cool001me@ssh-oidc-demo.data.kit.edu) Access Token:

cool001me@ssh-oidc-demo:~$
```

`flaat` + `pam-module` + `motley_cue`

https://motley-cue.readthedocs.io

- **`motley-cue`**: Server-side daemon developed in HIFIS (Germany)
- **`motley-cue`** fixes 4 things:
  1. Dynamically provision a user (plugin-based, **optional**)
     - Pooled-account, “Friendly” username, **External** username lookup
     - Authorisation based on
       - **`entitlement`** (i.e. VO)
       - **`assurance`**
       - **`sub@iss`** (user whitelist)
  2. If access token is longer than 1kb
     - **`motley_cue`** creates a one-time-password (OTP)
  3. Obtain the **username from server** for you
  4. Admin interface for **security incidents**
     - suspend / resume user
     - But opens a REST interface on the server

```
# Example
mccli ssh ssh-oidc-demo.data.kit.edu --oidc wlcg-demo

cool001@ssh-oidc-demo:~$
```

SSH notes

- No `ssh`-daemons (or clients) were hurt in this project:
  - **Unmodified SSH Client and Server**
  - **Backward compatible** with: password, ssh-keys, 2nd factor modules, ...
- **`mccli`**
  - wrapper around **unmodified client**:
  - oidc access token handling (via oidc-agent)
  - Currently supports:
    - ssh, scp
    - more to come
- Demo: https://ssh-oidc-demo.data.kit.edu
- Windows Client available: http://repo.data.kit.edu/windows/oidc-agent/
  - Use plugin of putty
  - Plugin supported by oidc-agent for windows available here
- Supported platforms
  - Windows: Putty
  - Mac/Linux: OpenSSH
- Packages: https://repo.data.kit.edu
- Video: https://youtu.be/090D4s0TNaA
- Visit https://ssh-oidc-demo.data.kit.edu to **try it yourself**

More SSH Approaches

- Multiple different approaches exist
- Smart Shell
  - AWI, SURF
- SSH Certificates
  - DEIC
- PAM Module
  - STFC, **KIT**

Important
We are working together
to make things **compatible**

That’s all

In case I talked too fast

Orpheus

https://orpheus.data.kit.edu

- Gain deep insights
  - into everything