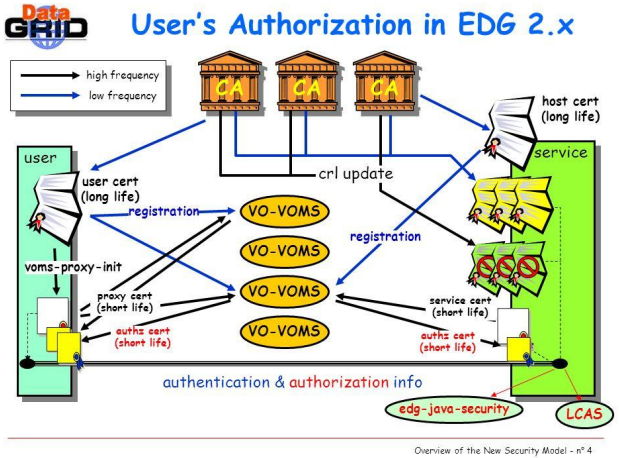
Da proxy VOMS a token: cosa cambia per gli utenti

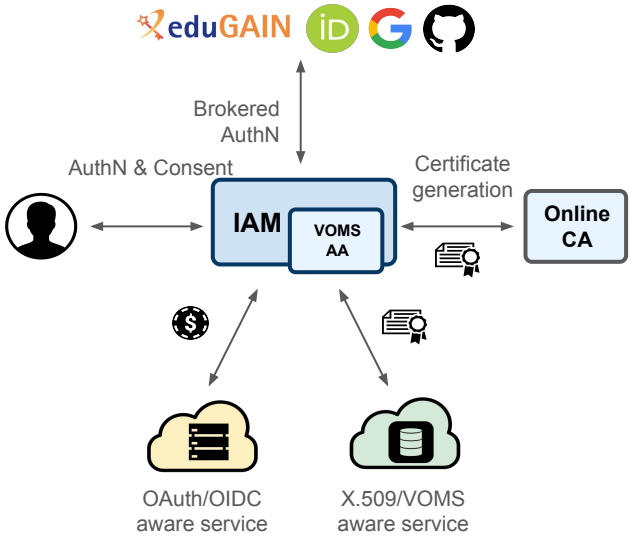
Corso di formazione “Panoramica su OAuth2/OpenID Connect e sue applicazioni tramite il servizio INDIGO IAM”, 12-14 Maggio 2025, LNF

Federica Agostini, INFN CNAF

Evolution of the WLCG AAI beyond X.509



Move beyond X.509



Approach: leverage and build upon the WLCG experience

X.509 certificates

Standard of the Internet Public Key Infrastructure (RFC 3280)

It is a **digital document** which links a public key to an identity

- a service/website, a machine, or a person

A certificate is associated to a private key

Issued and digitally signed by a trusted **Certificate Authority** (CA)

Long-lasting: typically, a certificate expires after one year and must be renewed

Authentication relies on the identity stated in the certificate

Authorization is based on information contained in ***extensions*** included in certificates derived from a personal certificate (*proxy*)

```
openssl x509 -in ~/.globus/usercert.pem -noout -text
```

```
Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:a8:c8:82:0d:1f:b6:1a:a8:a3:08:5b:89:08:e3:
                    51:34:80:3a:ad:69:02:98:29:d0:af:d5:c0:cb:f7:
                    f8:9b:83:44:0b:3b:f8:72:6a:4f:6e:f7:34:27:a7:
                    b5:e5:2b:a7:bd:37:09:cc:4f:77:9f:9f:7a:39:c5:
                    6c:80:e2:76:3c:51:6b:d7:41:1c:50:8e:e4:0e:d6:
                    1c:4c:6a:7c:45:c7:35:47:61:89:1d:e6:9e:09:0f:
                    2d:e5:ac:34:8f:83:0f:32:a2:33:d9:88:0c:15:70:
                    35:99:7d:5c:d7:10:2f:c4:10:0b:5b:85:e4:99:73:
                    27:50:68:45:92:08:b9:98:f6:ba:eb:8b:2b:f1:ac:
                    89:58:6a:20:33:5c:58:55:d2:14:6c:c4:fc:bd:16:
                    4f:39:9e:ad:49:57:37:80:37:b8:d5:b7:72:55:ac:
                    38:07:77:5e:c8:40:8c:92:5f:3d:c9:53:b5:18:0a:
                    11:9c:5b:70:8e:e0:b3:5e:e2:9d:fd:71:b8:d6:eb:
                    06:e2:52:aa:3e:ed:c6:92:26:c3:b1:c2:84:b6:45:
                    c1:c0:6a:18:1d:f8:11:9e:80:c9:2b:e3:c9:2b:9a:
                    88:ee:0d:c6:2b:ae:40:0b:e0:02:b4:52:1e:71:5d:
                    d5:ae:97:59:06:d1:21:5f:d1:6c:8d:db:70:a3:b8:
                    51:17
                Exponent: 65537 (0x10001)
```

Public key

```
X509v3 extensions:
        X509v3 Basic Constraints: critical
            CA:FALSE
```

Some extension

```
Issuer: C = IT, O = IGI, CN = Test CA
Validity
    Not Before: Oct  1 13:16:32 2022 GMT
    Not After : Sep 28 13:16:32 2032 GMT
Subject: C = IT, O = IGI, CN = test0
```

Issuer/Subject DNs and validity

```
Signature Algorithm: sha1WithRSAEncryption
Signature Value:
    47:06:4e:4f:72:bd:c4:5c:96:5e:74:f7:84:42:e5:c2:74:f3:
    03:00:9a:40:71:94:c2:4c:83:5f:19:1a:10:5c:36:10:93:a7:
    1e:f1:b1:fc:8c:73:e6:6a:8e:9d:a9:5f:df:ff:5e:32:d8:99:
    a4:cf:e6:a2:15:08:52:4b:20:cb:93:8c:b0:4e:01:87:4f:e9:
    29:58:4b:18:19:b3:78:01:d7:0c:94:7d:4b:f6:e6:86:ed:86:
    ee:01:9c:c7:e9:b9:1d:f0:72:ec:4e:2c:2c:95:76:07:8f:29:
    5d:6d:02:fd:65:4e:83:ac:60:1d:b7:f2:2e:01:f1:6b:42:35:
    96:b8:9d:b5:49:f0:65:9e:9e:0e:91:64:0a:e7:04:e4:b1:3b:
    df:79:95:fa:57:08:a7:a5:7c:c5:eb:12:2b:b4:07:c9:ec:e6:
    11:0e:e3:7b:50:8a:f0:06:5f:46:d2:1a:10:3e:e5:c5:25:e5:
    51:03:36:11:c0:35:5b:7a:34:a8:ff:c3:96:0a:3d:7b:11:70:
    ce:fa:d1:ba:31:df:89:60:8b:2a:27:8c:2b:ca:d7:78:61:eb:
    98:0d:50:f3:6d:55:7c:0c:77:cc:ae:4d:be:3d:e2:74:9d:23:
    92:c9:85:7e:76:86:85:cc:9e:bc:df:bb:f0:f3:ee:60:c4:34:
    c7:4f:fb:a7
```

CA digital signature

Proxy certificate

Standard of the X.509 Public Key Infrastructure (RFC 3820)

It is generated and signed by a user's certificate and contains

- **proxy certificate**
  - subject DN is derived by user's DN
  - includes the user's identity information
  - short lifetime (24/48 hours), limiting potential damage if compromised
- **private key**
  - no password required
  - accessible only by the owner
- **user's certificate**

Additional proxy certificates can be generated from an existing proxy and its private key, allowing for **delegation**

- e.g. a service can create a proxy certificate derived by a user proxy

A PEM textual representation of the binary certificate (in ASN.1 DER format)

```
-----BEGIN CERTIFICATE-----
MIIDGTCCAgGgAwIBAgIEG8PNrDANBgkqhkiG9w0BAQ0FADArMQswCQYDVQQGEwJJ
VDEMMAoGA1UECgwDSUdJMQ4wDAYDVQQDDAV0ZXN0MDAeFw0yNTA1MDUxNjUxMTha
Fw0yNTA1MDYwNDUxMThaMD8xCzAJBgNVBAYTAklUMQwwCgYDVQQKDANJR0kxDjAM
BgNVBAMMBXRlc3QwMRIwEAYDVQQDEwk0NjU4MTcwMDQwggEiMA0GCSqGSIb3DQEB
AQUAA4IBDwAwggEKAoIBAQCVG0p2bUn5BRzQrEtqgczReZHNybQcCusDN7q4dgeC
B2WjGKDdKEpyUHRfhesQcv9uMxF0Gi9ml2+ruWA=
-----END CERTIFICATE-----
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAlRtKdm1J+QUc0KxLaoHM0XmRzcm0HArrAze6uHYHgqQuWaWT
HTb87K+ThPLt+BGFBs9bepkXNMM6DXKhf8CiydnNUw7hMkmfEM6AMwtFX2Fu70zx
DfMUONq7W8mWcVq3ugxcV9t4P9uvu0ay/maweCEEduo6/PwcamIqnIl/vJ7i9Zd/
XlaBg3R7+VkQV2K7boD1zFMOL5SmNh4KoWlu9oSBM4lPUV8zw5cYJJkibG2LOrwF
cNkixOG5xn8+ufA+BKvhd/x7oC/6sstP7F4aMlk1F1q90THNdsRGldm1Kxg9JQl0
k6/BCUnVFMh0MNnWSANF0CvgMgtq5uM8DQmn4Hp3qlhEqVhLSeZHgwQvTE90qtjK
4DCHpekCgYBuZdDw8ls8uacvqdMLy+R3C5248fcDFx8siPpFp8uVXXUntRH5nVAX
h3O542Flf7kOo0arUtOAkFjcovU4Po/ysvldyNnZxARNiSOXUsbzogynECY0F9es
t0cxuCNLMrl3BveXk6aeWJ5evnBfeb5eD1M7X5Tv9dQyNLuxNoS34A==
-----END RSA PRIVATE KEY-----
```

```
-----BEGIN CERTIFICATE-----
MIIDZTCCAk2gAwIBAgIUc0EdbhdwosQZCWnL5phnLmhhzEUwDQYJKoZIhvcNAQEN
Af8EAjAAMB0GA1UdDgQWBBTqxeHhLdB96TLXS7L0uZyWhgaNBTAOBgNVHQ8BAf8E
BAMCBeAwHwYDVR0jBBgwFoAUepsdRY+opu/tuVsDK27KNiuhfrcwHQYDVR0RBBYw
FIESdGVzdDBAY25hZi5pbmZuLml0MA0GCSqGSIb3DQEBDQUAA4IBAQB6SYZKs2uP
GGprpV+l1lF1YPFP5v9TZL3GiJhSeA4asLL/NZwVVTfG82Iw+OB2HTbCA6FRXPER
6t6S0uDgva6IWKfBxvR3liQd7WOC/TfQ8iYkq0XuPIWIA/zcYXXlnkssY01WGXnu
HLk2rMyD+vZfhr/ISZ83q73qfCnL4lUrXP5cl9yvG4n5Rf0h+Cg7eLTIn5xqrfWz
cpojlu1I6Hk4thwzDQYmk7Vs3/5iOrhoWwE6be8bvFfRgXkOaddJbCcnGyHP23Kz
cSIciUQ0NCR6
-----END CERTIFICATE-----
```

VOMS proxy

Accessing a resource requires proving both identity and membership to a Virtual Organization (VO)

A **VOMS proxy** is a proxy certificate that includes a VO-specific **Attribute Certificate (AC)**

To create a VOMS proxy, the user contacts a VOMS server, which issues an AC with VO membership, group, and role information

Grid services may take **authorization decisions** based on VOMS extensions, e.g.

- can the user read this file?
- is the user allowed to submit a job?

```
voms-proxy-info -all
```

```
subject   : /C=IT/O=IGI/CN=test0/CN=1629861880
issuer    : /C=IT/O=IGI/CN=test0
identity  : /C=IT/O=IGI/CN=test0
type      : RFC3820 compliant impersonation proxy
strength  : 2048
path      : /tmp/x509up_u1000
timeleft  : 11:59:52
key usage : Digital Signature, Non Repudiation, Key Encipherment
=== VO indigo-dc extension information ===
VO        : indigo-dc
subject   : /C=IT/O=IGI/CN=test0
issuer    : /C=IT/O=IGI/CN=voms.test.example
attribute : /indigo-dc/Role=NULL/Capability=NULL
attribute : /indigo-dc/xfers/Role=NULL/Capability=NULL
timeleft  : 11:59:52
uri       : voms.test.example:8080
```

Proxy certificates - pros and cons

Pros

Extension of a secure and well-established authentication technology

- X.509 certificates are (primarily) used on the server side
- proxy certificates allow for client-side authentication

Cons

- Usability challenges
  - obtaining and managing X.509 credentials can be complex (e.g. installation on different machines)
- Lack of adoption outside our specialized domains
  - for example, web browsers and servers do not natively support this technology

JSON Web Token (JWT)

More recent, standard technology (RFC 7519)

- common in Web environment

JSON Web Token is a compact, self-contained way of securely transmitting information between parties in a JSON object

The payload of the JWT is base64-url encoded.
It contains claims, which hold

- user identity information (e.g. membership to groups), and/or
- capabilities, in the form of OAuth *scopes*

JWTs are typically **signed** by the token issuer

JWTs are usually short-lived (about **1h**)

A JWT is represented as a sequence of **URL-safe parts** separated by period (“.”) characters

JWT: Header.Body.Signature

Example of encoded token

```
eyJraWQiOiJyc2ExIiwiYWxnIjoiUlMyNTYifQ.eyJ3bGNnLnZlciI6IjEuMCIsInN1YiI6IjgwZTVmYjhkLWI3YzgtNDUxYS04OWJhLTM0NmFlMjc4YTY2ZiIsImF1ZCI6Imh0dHBzOlwvXC93bGNnLmNlcm4uY2hcL2p3dFwvdjFcL2FueSIsIm5iZiI6MTc0NjQ3MjkxMiwic2NvcGUiOiJvcGVuaWQgb2ZmbGluZV9hY2Nlc3Mgd2xjZy5ncm91cHMiLCJpc3MiOiJodHRwczpcL1wvaWFtLnRlc3QuZXhhbXBsZVwvIiwiZXhwIjoxNzQ2NDc2NTEyLCJpYXQiOjE3NDY0NzI5MTIsImp0aSI6IjQyYWUzNjY5LTIyYjctNDEyOC05MDM3LTFlMGY2MTk5MmIyYiIsImNsaWVudF9pZCI6IjZhODY3MTdiLTUxNTMtNDU5Mi1hNjM2LTJiZjAyMTY5NGE1OCIsIndsY2cuZ3JvdXBzIjpbIlwvQW5hbHlzaXMiLCJcL1byb2R1Y3Rpb24iLCJcL2luZGlnby1kYyIsIlwvaW5kaWdvLWRjXC94ZmVycyJdfQ.TeUM-nM9cwgiWlyAJVZwaWlNLkoY2A6n-ekt0c6GIxOpzqifzN6VUvm0xFY30K-rrhhsay0rGnRqqZoPF5S5IWJwPiSovsCysmxH5U3LYav0BeT8MpEToFArDtFMyWrzvW2pACmxCWjamFWahhteTAcD3UcMOfd6sddjPj4kIW0zsdT2rCmRshRLiU_NFBu5qFKpRIfn7svGS_nkR-0pfOW2hjSBKHtufWIgGrBFv0Y6Qu5k70lENK2us8Drrnj9WXuOvZJny32h3_3LinpKbXvW0Cfke3ddvYjJCVepk3U4Sij-wSq2AwjvhO1mXffb9M07CCYsOxtB6shk6DxU2g
```

Example of decoded token

```
{
  "kid": "rsa1",
  "alg": "RS256"
}
{
  "wlcg.ver": "1.0",
  "sub": "80e5fb8d-b7c8-451a-89ba-346ae278a66f",
  "aud": "https://wlcg.cern.ch/jwt/v1/any",
  "nbf": 1746472912,
  "scope": "openid offline_access wlcg.groups",
  "iss": "https://iam.test.example/",
  "exp": 1746476512,
  "iat": 1746472912,
  "jti": "42ae3669-22b7-4128-9037-1e0f61992b2b",
  "client_id": "6a86717b-5153-4592-a636-2bf021694a58",
  "wlcg.groups": [
       "/Analysis",
       "/Production",
       "/indigo-dc",
       "/indigo-dc/xfers"
  ]
}
TeUM-nM9cwgiWlyAJVZwaWlNLkoY2A6n-ekt0c6GIxOpzqifzN6VUvm0xFY30K-rrhhsay0rGnRqqZoPF5S5IWJwPiSovsCysmxH5U3LYav0BeT8MpEToFArDtFMyWrzvW2pACmxCWjamFWahhteTAcD3UcMOfd6sddjPj4kIW0zsdT2rCmRshRLiU_NFBu5qFKpRIfn7svGS_nkR-0pfOW2hjSBKHtufWIgGrBFv0Y6Qu5k70lENK2us8Drrnj9WXuOvZJny32h3_3LinpKbXvW0Cfke3ddvYjJCVepk3U4Sij-wSq2AwjvhO1mXffb9M07CCYsOxtB6shk6DxU2g
```

OAuth Bearer Token usage

Defined in RFC 6750

It describes how to use tokens in HTTP requests to access resources

- here we mean OAuth Resource Servers

Any Client in possession of a Bearer token (a *bearer*) can use it to **get access** to the associated resources

Typically, tokens are sent in the **Authorization HTTP header**

```
GET /fga/fga_testing_curl HTTP/1.1
Host: storm.test.example:8443
User-Agent: curl/7.76.1
Accept: */*
Authorization: Bearer
eyJraWQiOiJyc2ExIiwiYWxnIjoiUlMyNTYifQ.eyJ3bGNnLnZlciI6
IjEuMCIsInN1YiI6IjgwZTVmYjhkLWI3YzgtNDUxYS04OWJhLTM0NmF
lMjc4YTY2ZiIsImF1ZCI6Imh0dHBzOlwvXC93bGNnLmNlcm4uY2hcL2
p3dFwvdjFcL2FueSIsIm5iZiI6MTc0NjQ3MjkxMiwic2NvcGUiOiJvc
GVuaWQgb2ZmbGluZV9hY2Nlc3Mgd2xjZy5ncm91cHMiLCJpc3MiOiJo
dHRwczpcL1wvaWFtLnRlc3QuZXhhbXBsZVwvIiwiZXhwIjoxNzQ2NDc
2NTEyLCJpYXQiOjE3NDY0NzI5MTIsImp0aSI6IjQyYWUzNjY5LTIyYj
ctNDEyOC05MDM3LTFlMGY2MTk5MmIyYiIsImNsaWVudF9pZCI6IjZhO
DY3MTdiLTUxNTMtNDU5Mi1hNjM2LTJiZjAyMTY5NGE1OCIsIndsY2cu
Z3JvdXBzIjpbIlwvQW5hbHlzaXMiLCJcL1byb2R1Y3Rpb24iLCJcL2l
uZGlnby1kYyIsIlwvaW5kaWdvLWRjXC94ZmVycyJdfQ.TeUM-nM9cwg
iWlyAJVZwaWlNLkoY2A6n-ekt0c6GIxOpzqifzN6VUvm0xFY30K-rrh
hsay0rGnRqqZoPF5S5IWJwPiSovsCysmxH5U3LYav0BeT8MpEToFArD
tFMyWrzvW2pACmxCWjamFWahhteTAcD3UcMOfd6sddjPj4kIW0zsdT2
rCmRshRLiU_NFBu5qFKpRIfn7svGS_nkR-0pfOW2hjSBKHtufWIgGrB
Fv0Y6Qu5k70lENK2us8Drrnj9WXuOvZJny32h3_3LinpKbXvW0Cfke3
ddvYjJCVepk3U4Sij-wSq2AwjvhO1mXffb9M07CCYsOxtB6shk6DxU2
g
```

JWTs - pros and cons

Pro

- Widely adopted industry standard
- Well integrated into the web ecosystem

Cons

- Driven by external standards
  - limited control over its evolution
- Still evolving
  - specifications and best practices may change
- Challenging to apply outside the web domain

Command line comparison

The next examples are taken from this tutorial

Client setup

VOMS proxies	JWTs
Obtain a X.509 certificate from a CA	–
Import the certificate in the browser	–
Registration to VOMS	Registration to IAM
Copy the user certificate and private key into an UI (*)	–
Set proper permissions to the private key file	–
–	Register a Client into IAM and save its configuration locally
Use `voms-proxy-init` to get a proxy	Use tools like `oidc-agent` (or directly `curl`) to get a token
Use Grid clients (e.g. `Gfal`) to access resources	Use Grid clients (e.g. `Gfal`, or directly `curl`) to access resources

(*) alternatively, install voms clients locally, create LSC and vomses files, install IGTF certificates locally (in a well-known path)

Register an OAuth Client with oidc-agent (only once)

This may be done by any UI, also locally!

```
$ eval $(oidc-agent)
$ oidc-gen -w device test0
[1] https://wlcg.cloud.cnaf.infn.it/
[2] https://iam-dev.cloud.cnaf.infn.it/
…
Issuer [https://iam-demo.cloud.cnaf.infn.it/]: https://iam.test.example/
The following scopes are supported: openid profile email offline_access wlcg wlcg.groups storage.read:/ storage.create:/
compute.read compute.modify compute.create compute.cancel storage.modify:/ eduperson_scoped_affiliation
eduperson_entitlement eduperson_assurance storage.stage:/
Scopes or 'max' (space separated) [openid profile offline_access]: openid profile offline_access wlcg.groups:/
Registering Client ...
Generating account configuration ...
accepted

Using a browser on any device, visit:
https://wlcg.cloud.cnaf.infn.it/device

And enter the code: LYE0TT

[ Polling the device code verification from the https://iam.test.example/device/approve endpoint ]

Enter encryption password for account configuration 'demo': ***
Confirm encryption Password: ***
Everything setup correctly!
```

oidc-agent saves the Client configuration (id, secret, etc.) and a refresh token locally. The configuration is protected with password

Get a token with oidc-agent

```
$ eval $(oidc-agent)
Agent pid 33
$ oidc-add test0
Enter decryption password for account config 'test0':
success
$ oidc-token test0
eyJraWQiOiJyc2ExIiwiYWxnIjoiUlMyNTYifQ.eyJ3bGNnLnZlciI6IjEuMCIsInN1YiI6IjgwZTVmYjhkLWI3YzgtNDUxYS04OWJhLTM0NmFlMjc4YTY2ZiIsImF1ZCI6Imh0dHBzOlwvXC93bGNnLmNlcm4uY2hcL2p3dFwvdjFcL2FueSIsIm5iZiI6MTc0NjM4OTE1Niwic2NvcGUiOiJvcGVuaWQgb2ZmbGluZV9hY2Nlc3Mgd2xjZy5ncm91cHMiLCJpc3MiOiJodHRwczpcL1wvaWFtLnRlc3QuZXhhbXBsZVwvIiwiZXhwIjoxNzQ2MzkyNzU2LCJpYXQiOjE3NDYzODkxNTYsImp0aSI6ImFlMDg3M2RlLTFhODQtNDA5MS04ZWUyLWExNzZmZTI2MTEyYyIsImNsaWVudF9pZCI6IjZhODY3MTdiLTUxNTMtNDU5Mi1hNjM2LTJiZjAyMTY5NGE1OCIsIndsY2cuZ3JvdXBzIjpbIlwvQW5hbHlzaXMiLCJcL1byb2R1Y3Rpb24iLCJcL2luZGlnby1kYyIsIlwvaW5kaWdvLWRjXC94ZmVycyJdfQ.JSx0fJkECvl-4CreketiCOYP3goGfbFU_DPSktGJ9aPz60osVvi6RIvJShsX1F2F-x0RA6rT_JUaNRPt0MRK6tSsq-j-UuBfKybtX-EfEhxFEs1lw6zKUfnshY-Jfrs9TmHnhJxV05TGXxL9ssLVuZMvHORFTifcO2mGJZ3KGBtRw_roGV75DcHJdVUNXbxUrFh_MXVFOHdzCeBAinggP-EcLc4ZgxyuWFytAMEKesjYAD9oDpLAl5kCbKjj8TLDHKMDsApnfs0BhGWn4QQDbkgZraD4emdh030wEKxXt5d9EAqeO9rzQ4gaUg8ijAxSD3EzFsvvgIMO9tOfXLtjQQ
```

To be done only once per login session

Get a token with oidc-agent

```
$ eval $(oidc-agent)
Agent pid 33
$ oidc-add test0
Enter decryption password f
success
$ echo $(oidc-token test0)
```


eyJraWQiOiJyc2ExIiwiYWxnIj … WJhLTM0NmFlMjc4YTY2ZiIsImF1Z
CI6Imh0dHBzOlwvXC93bGNnLmN … 2ZmbGluZV9hY2Nlc3Mgd2xjZy5nc
m91cHMiLCJpc3MiOiJodHRwczp … TYsImp0aSI6ImFlMDg3M2RlLTFhO
DQtNDA5MS04ZWUyLWExNzZmZTI … CIsIndsY2cuZ3JvdXBzIjpbIlwvQ
W5hbHlzaXMiLCJcL1Byb2R1Y3R … tiCOYP3goGfbFU_DPSktGJ9aPz60
osVvi6RIvJShsX1F2F-x0RA6rT … LVuZMvHORFTifcO2mGJZ3KGBtRw_
roGV75DcHJdVUNXbxUrFh_MXVF … QDbkgZraD4emdh030wEKxXt5d9EA
qeO9rzQ4gaUg8ijAxSD3EzFsvv


```
$ echo $(oidc-token test0) | cut -d. -f2 | base64 -d 2>/dev/null | jq .
```


```
{
  "wlcg.ver": "1.0",
  "sub": "80e5fb8d-b7c8-451a-89ba-346ae278a66f",
  "aud": "https://wlcg.cern.ch/jwt/v1/any",
  "nbf": 1746389156,
  "scope": "openid offline_access wlcg.groups",
  "iss": "https://iam.test.example/",
  "exp": 1746392756,
  "iat": 1746389156,
  "jti": "ae0873de-1a84-4091-8ee2-a176fe26112c",
  "client_id": "6a86717b-5153-4592-a636-2bf021694a58",
  "wlcg.groups": [
       "/Analysis",
       "/Production",
       "/indigo-dc",
       "/indigo-dc/xfers"
  ]
}
```

Get a token with curl

```
$ curl -sfL -u ${IAM_CLIENT_ID}:${IAM_CLIENT_SECRET} -d client_id=${IAM_CLIENT_ID} -d scope="wlcg.groups"
https://iam.test.example/devicecode > /tmp/resp
$  cat /tmp/resp | jq .
{
  "user_code": "GDFWJL",
  "device_code": "44856603-0be0-41bf-b034-c456ade382b7",
  "verification_uri_complete": "https://iam.test.example/device?user_code=GDFWJL",
  "verification_uri": "https://iam.test.example/device",
  "expires_in": 600
}
$ AT=$(curl -sL -u ${IAM_CLIENT_ID}:${IAM_CLIENT_SECRET} -d grant_type=urn:ietf:params:oauth:grant-type:device_code -d
device_code=$(jq -r .device_code /tmp/resp) https://iam.test.example/token | jq .access_token | tr -d '"')
$ echo $AT
eyJraWQiOiJyc2ExIiwiYWxnIjoiUlMyNTYifQ.eyJ3bGNnLnZlciI6IjEuMCIsInN1YiI6IjgwZTVmYjhkLWI3YzgtNDUxYS04OWJhLTM0NmFlMjc4YTY2ZiIsImF1Z
CI6Imh0dHBzOlwvXC93bGNnLmNlcm4uY2hcL2p3dFwvdjFcL2FueSIsIm5iZiI6MTc0NjQ1NjU1OSwic2NvcGUiOiJ3bGNnLmdyb3VwcyIsImlzcyI6Imh0dHBzOlwvX
C9pYW0udGVzdC5leGFtcGxlXC8iLCJleHAiOjE3NDY0NjAxNTksImlhdCI6MTc0NjQ1NjU1OSwianRpIjoiODU4ZWFhMjMtMjY4MC00OWQ4LWE5ZjUtZDg1NGUyMWMyZ
DNmIiwiY2xpZW50X2lkIjoiNmE4NjcxN2ItNTE1My00NTkyLWE2MzYtMmJmMDIxNjk0YTU4Iiwid2xjZy5ncm91cHMiOlsiXC9BbmFseXNpcyIsIlwvUHJvZHVjdGlvb
iIsIlwvaW5kaWdvLWRjIiwiXC9pbmRpZ28tZGNcL3hmZXJzIl19.JQ7xYnOKJbWkhRQ_TAM0PuzmRQo5XrRCUH_Ze_d7tRgnfDzOSIpxOVvkOm54IrEbT7YIHKUekaqD
JqmrsesitLoRwOgkFnvR9mVol6x5LZBYqJ4_FoGI_ZljPcJ5kRtDnMujcV8Vd6ytlozgb1go0GZNURrfOIKmNlkgkEam-gyOXc2usBZ5D6_NJ-mb9pG_FPBId1GJ0_oI
3tGqjJvFoxPpNrpT7jTcdAGLoKrjlEwmZbowKqspUXhYuiViKCRXVxMwYBswdQUjlBnR-o1ak3wEWXDTWgrJvk75AoFw1J3jkapP_1jzOFiEdQ8b4dOk0ijRu26os8rN
HUbd02voYQ
```

Browse this url and approve the client

Get a token with curl

```
$ curl -sLf -u ${IAM_CLIENT_ID}:${IAM_CLIENT_SECRET} -d client_id=${IAM_CLIENT_ID} -d scope="wlcg.groups"
https://iam.test.example/devicecode > /tmp/resp
$  cat /tmp/resp | jq .
{
  "user_code": "GDFWJL",
  "device_code": "44856603-0be0-4
  "verification_uri_complete": "h
  "verification_uri": "https://ia
  "expires_in": 600
}
$ AT=$(curl -sL -u ${IAM_CLIENT_I                                                    grant-type:device_code -d
device_code=$(jq -r .device_code                                                     | tr -d '"')
$ echo $AT
eyJraWQiOiJyc2ExIiwiYWxnIjoiUlMyN                                                    UxYS04OWJhLTM0NmFlMjc4YTY2ZiIsImF1Z
CI6Imh0dHBzOlwvXC93bGNnLmNlcm4uY2                                                    NnLmdyb3VwcyIsImlzcyI6Imh0dHBzOlwvX
C9pYW0udGVzdC5leGFtcGxlXC8iLCJleH                                                   MtMjY4MC00OWQ4LWE5ZjUtZDg1NGUyMWMyZ
DNmIiwiY2xpZW50X2lkIjoiNmE4NjcxN2                                                   siXC9BbmFseXNpcyIsIlwvUHJvZHVjdGlvb
iIsIlwvaW5kaWdvLWRjIiwiXC9pbmRpZ2                                                   RgnfDzOSIpxOVvkOm54IrEbT7YIHKUekaqD
JqmrsesitLoRwOgkFnvR9mVol6x5LZBYq                                                   gyOXc2usBZ5D6_NJ-mb9pG_FPBId1GJ0_oI
3tGqjJvFoxPpNrpT7jTcdAGLoKrjlEwmZ                                                   U3jkapP_1jzOFiEdQ8b4dOk0ijRu26os8rN
HUbd02voYQ
```

```
$ echo $AT | cut -d. -f2 | base64 -d 2>/dev/null | jq .
{
  "wlcg.ver": "1.0",
  "sub": "80e5fb8d-b7c8-451a-89ba-346ae278a66f",
  "aud": "https://wlcg.cern.ch/jwt/v1/any",
  "nbf": 1746456559,
  "scope": "wlcg.groups",
  "iss": "https://iam.test.example/",
  "exp": 1746460159,
  "iat": 1746456559,
  "jti": "858eaa23-2680-49d8-a9f5-d854e21c2d3f",
  "client_id": "6a86717b-5153-4592-a636-2bf021694a58",
  "wlcg.groups": [
      "/Analysis",
      "/Production",
      "/indigo-dc",
      "/indigo-dc/xfers"
  ]
}
```

Groups and roles

Both **INDIGO IAM** and **VOMS** support the concepts of *groups* and *roles*

Primary groups

- **VOMS:** the order of the requested group will be the same as the one appearing in the proxy
- **INDIGO IAM**: the same can be obtained for WLCG tokens while requesting `wlcg.groups` in a certain order

Roles

- **VOMS**: they have to be explicitly requested (also multiple times)
- **INDIGO IAM**: the same behavior is obtained through *optional groups* for WLCG JWT tokens (the list of user's role memberships appears in the `wlcg.groups` claim)

In IAM, a VOMS *role* (also called *optional group*) is represented by the `voms.role` and `wlcg.optional-group` labels

Client request for a (primary) group

VOMS proxy with **/indigo-dc/xfers** attribute set as primary group

```
$ echo pass | voms-proxy-init -voms indigo-dc -order
/indigo-dc/xfers -pwstdin
$ voms-proxy-info -all
subject   : /C=IT/O=IGI/CN=test0/CN=1499357201
issuer    : /C=IT/O=IGI/CN=test0
identity  : /C=IT/O=IGI/CN=test0
type      : RFC3820 compliant impersonation proxy
strength  : 2048
path      : /tmp/x509up_u1000
timeleft  : 11:57:51
key usage : Digital Signature, Non Repudiation, Key
Encipherment
=== VO indigo-dc extension information ===
VO        : indigo-dc
subject   : /C=IT/O=IGI/CN=test0
issuer    : /C=IT/O=IGI/CN=voms.test.example
attribute : /indigo-dc/xfers/Role=NULL/Capability=NULL
attribute : /indigo-dc/Role=NULL/Capability=NULL
timeleft  : 11:57:51
uri       : voms.test.example:8080
```

Access token with **/indigo-dc/xfers** default group set as primary

```
$ AT=$(oidc-token -s wlcg.groups:/indigo-dc/xfers test0)
$ echo $AT | cut -d. -f2 | base64 -d 2>/dev/null | jq .
{
  "wlcg.ver": "1.0",
  "sub": "80e5fb8d-b7c8-451a-89ba-346ae278a66f",
  "aud": "https://wlcg.cern.ch/jwt/v1/any",
  "nbf": 1746377883,
  "scope": "wlcg.groups:/indigo-dc/xfers",
  "iss": "https://iam.test.example/",
  "exp": 1746381483,
  "iat": 1746377883,
  "jti": "ffc95ac6-114b-44fc-bbc6-5bc78382b63e",
  "client_id": "6a86717b-5153-4592-a636-2bf021694a58",
  "wlcg.groups": [
    "/indigo-dc/xfers",
    "/Analysis",
    "/Production",
    "/indigo-dc"
  ]
}
```

Client request for a VOMS Role/optional group

VOMS proxy with **webdav** Role

```
$ echo pass | voms-proxy-init -voms
indigo-dc:/indigo-dc/Role=webdav -pwstdin
$ voms-proxy-info -all
subject   : /C=IT/O=IGI/CN=test0/CN=2077821009
issuer    : /C=IT/O=IGI/CN=test0
identity  : /C=IT/O=IGI/CN=test0
type      : RFC3820 compliant impersonation proxy
strength  : 2048
path      : /tmp/x509up_u1000
timeleft  : 11:59:54
key usage : Digital Signature, Non Repudiation, Key
Encipherment
=== VO indigo-dc extension information ===
VO        : indigo-dc
subject   : /C=IT/O=IGI/CN=test0
issuer    : /C=IT/O=IGI/CN=voms.test.example
attribute : /indigo-dc/Role=webdav/Capability=NULL
attribute : /indigo-dc/Role=NULL/Capability=NULL
attribute : /indigo-dc/xfers/Role=NULL/Capability=NULL
timeleft  : 11:59:54
uri       : voms.test.example:8080
```

Access token with **/indigo-dc/webdav** optional group

```
$ AT=$(oidc-token -s wlcg.groups:/indigo-dc/webdav test0)
$ echo $AT | cut -d. -f2 | base64 -d 2>/dev/null | jq .
{
  "wlcg.ver": "1.0",
  "sub": "80e5fb8d-b7c8-451a-89ba-346ae278a66f",
  "aud": "https://wlcg.cern.ch/jwt/v1/any",
  "nbf": 1746384391,
  "scope": "wlcg.groups:/indigo-dc/webdav",
  "iss": "https://iam.test.example/",
  "exp": 1746387991,
  "iat": 1746384391,
  "jti": "e4234ed7-deb8-48c9-bd39-2d0f59c9f575",
  "client_id": "6a86717b-5153-4592-a636-2bf021694a58",
  "wlcg.groups": [
    "/indigo-dc/webdav",
    "/Analysis",
    "/Production",
    "/indigo-dc",
    "/indigo-dc/xfers"
  ]
}
```

In IAM, **/indigo-dc/webdav** is a group tagged as *optional*

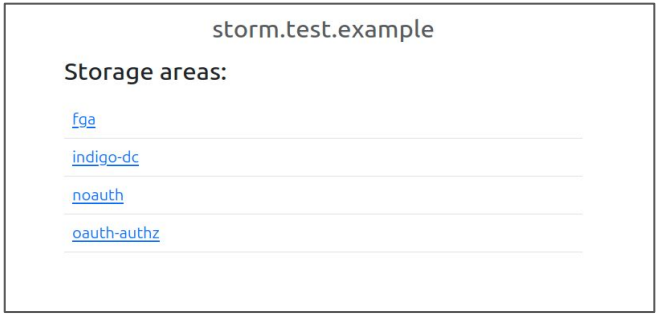
Resource setup: StoRM WebDAV

The StoRM WebDAV service provides HTTP/WebDAV access to resources on a shared filesystem, here used to showcase the access with VOMS proxies and JWTs

StoRM WebDAV has been configured to serve the `fga` Storage Area (SA) with the following permissions

- read/write access to members of the IAM `/indigo-dc/webdav` optional group or VOMS Role = `webdav`
- read access to tokens issued by https://iam.test.example and proxies signed by the `indigo-dc` VO

https://storm.test.example:8443/



The StoRM WebDAV documentation is available here

Transfer with proxy

Read/write access to `fga` with VOMS Role **`webdav`**

```
$ echo pass | voms-proxy-init -voms indigo-dc:/indigo-dc/Role=webdav -pwstdin
$ echo "Testing the fine grained SA permissions" > fga_testing
$ gfal-copy fga_testing https://storm.test.example:8443/fga/fga_testing
Copying file:///home/test/fga_testing   [DONE]  after 0s
$ gfal-cat https://storm.test.example:8443/fga/fga_testing
Testing the fine grained SA permissions
```

Read-only access to `fga` with VOMS **`indigo-dc`** VO

```
$ echo pass | voms-proxy-init -voms indigo-dc -pwstdin
$ gfal-cat https://storm.test.example:8443/fga/fga_testing
Testing the fine grained SA permissions
$ gfal-rm https://storm.test.example:8443/fga/fga_testing
https://storm.test.example:8443/fga/fga_testing FAILED
gfal-rm error: 1 (Operation not permitted) - DavPosix::unlink  HTTP 403 : Permission refused
```

`gfal` looks for a proxy in `/tmp/x509up_u<id -u>`

Transfer with token

Read/write access to `fga` with optional group **`webdav`**

```
$ export BEARER_TOKEN=$(oidc-token -s wlcg.groups:/indigo-dc/webdav test0)
$ echo "Testing the fine grained SA permissions" > fga_testing
$ gfal-copy fga_testing https://storm.test.example:8443/fga/fga_testing
Copying file:///home/test/fga_testing   [DONE]  after 0s
$ gfal-cat https://storm.test.example:8443/fga/fga_testing
Testing the fine grained SA permissions
```

Read-only access to `fga` with tokens issued by IAM

```
$ export BEARER_TOKEN=$(oidc-token test0)
$ gfal-cat https://storm.test.example:8443/fga/fga_testing
Testing the fine grained SA permissions
$ gfal-rm https://storm.test.example:8443/fga/fga_testing
https://storm.test.example:8443/fga/fga_testing FAILED
gfal-rm error: 1 (Operation not permitted) - DavPosix::unlink  HTTP 403 : Permission refused
```

gfal looks for a token saved in the **`BEARER_TOKEN`** environment variable, as defined in the WLCG Token Usage and Discovery document

Transfer with token using curl

Read/write access to `fga` with optional group **`webdav`**

```
$ AT=$(oidc-token -s wlcg.groups:/indigo-dc/webdav test0)
$ echo "Testing the fine grained SA permissions" > fga_testing
$ curl -H "Authorization: Bearer $AT" https://storm.test.example:8443/fga/fga_testing_curl -XPUT
--upload-file fga_testing -w '%{http_code}\n'
201
$ curl -H "Authorization: Bearer $AT" https://storm.test.example:8443/fga/fga_testing_curl
Testing the fine grained SA permissions
```

Read-only access to `fga` with tokens issued by IAM

```
$ AT=$(oidc-token test0)
$ curl -H "Authorization: Bearer $AT" https://storm.test.example:8443/fga/fga_testing_curl
Testing the fine grained SA permissions
$ curl -H "Authorization: Bearer $AT" https://storm.test.example:8443/fga/fga_testing_curl -XDELETE -w
'%{http_code}\n' -s -o /dev/null
403
```

The JWT is sent to StoRM WebDAV in the *Authorization* header

Hands-on

Hands-on

The previous examples have been performed through a self-contained deployment with docker-compose, including a

- `vomsaa` service which issues VOMS proxies
- `iam` service which issues JWTs
- `webdav` service to give access to certain resources

The repo is available here

Follow the README for more tutorials!

StoRM WebDAV authZ

StoRM WebDAV fine-grained authorization documentation here

The StoRM WebDAV service provides HTTP/WebDAV access to resources on a shared filesystem, here used to showcase the access with VOMS proxies and JWTs

StoRM WebDAV has been configured to serve the following Storage Areas (SA):

- **`fga`**: fine-grained authZ SA, whose policies are described here
  - **`.`**:
    - read/write access to members of the IAM **`/indigo-dc/webdav`** optional group or VOMS Role = **`webdav`**
    - read access to tokens issued by IAM and proxies signed by the **`indigo-dc`** VO
  - **`xfers:`** read/write access to members of the IAM **`/indigo-dc/xfers`** default group or VOMS group = **`webdav`**
  - **`public`**: read access to anyone
- **`indigo-dc`**: read/write access to **`indigo-dc`** VOMS proxies
- **`oauth-authz`**: read/write access to tokens issued by IAM
- **`noauth`**: access allowed without authentication

Folder tree