OAuth based AuthN/ AuthZ in DIRAC

A.Lytovchenko,
A.Tsaregorodtsev,
CPPM-IN2P3-CNRS, Marseille,
9th DIRAC User Workshop,
16 May 2019, London

Disclaimer

- Most of the work is done by Andrii Lytovchenko
- Questions on technical details of the OAuthDIRAC implementation are to be redirected to him

Outline

- Current DIRAC security framework
- Enabling DIRAC to use OAuth AAI
- OAuthDIRAC extension
- OAuthManager service
- X509 Proxy Providers
- Status and plans
- Conclusions

X509 based security

- DIRAC is using X509 certificates for user authentication
- Certificate proxy delegation protocol is used to pass the user credentials to remote components performing operations on behalf of the users
- User rights are determined by the group membership encoded in the DIRAC proxy extension
- The ProxyManager stores long-living user proxies in the ProxyDB and serves short (limited) proxies to the components operating on behalf of the user

DIRAC and VOMS

- DIRAC users are members of at least one VO managed by a VOMS service
- User rights defined in VOMS as groups and roles are translated into DIRAC group membership
  - VOMS synchronization with VOMS2CSAgent
- DIRAC proxies can be dressed with VOMS extensions to access external grid services
  - VOMS and DIRAC proxy extension coexist in the same proxy

X509 -> OIDC/OAuth

- Using X509 certificates is complicated for the end-users
  - Complex issuing procedure, yearly renewal, installation in multiple places with a format conversion, loading in browsers, etc, etc
  - Users of many communities do not have access to Certification Authorities issuing X509 certificates
- Need for a new non-X509 security infrastructure
  - Industry standard
  - Widely accepted
- OAuth2.0 + OIDC is the suitable choice
  - Although not a single one

OAuth/OIDC/SSO

- **OAuth 2.0** is the industry-standard *delegation* protocol for conveying *authorization decisions* across a network of web-enabled applications and APIs

- **O**pen **ID C**onnect – is an identity layer on top of the **OAuth 2.0**.
  - allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server

- Single sign-on (**SSO**) is an authentication process that allows a user to access multiple applications with one set of login credentials.

SSO solutions

- There are multiple examples of SSO solutions
- The EGI Check-in service enables access to EGI services and resources using federated authentication mechanisms
  - A hub between federated Identity Providers (IdPs) and Service Providers (SPs) that are part of EGI

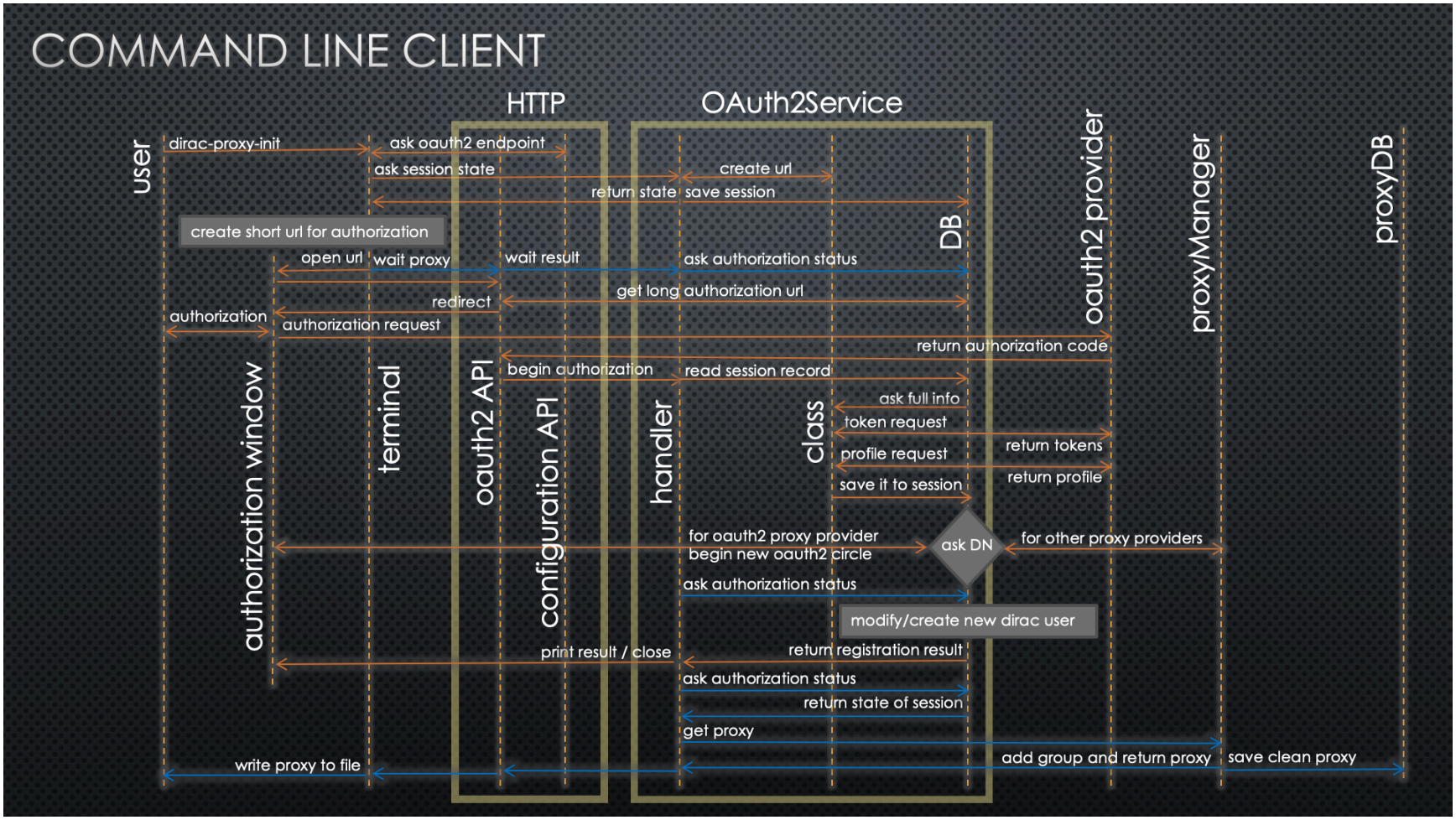
DIRAC support for SSO authentication

- Authenticate DIRAC users with the help of an external Authentication server
  - E.g. delegate it to EGI Check-In
- Get user profile information and eventually register users in DIRAC for supported VO's
  - Put users into DIRAC groups corresponding to the user profile
  - Similar to the procedure of synchronization with VOMS
- Ensure provisioning of X509 certificate proxies to be used for internal DIRAC client-server communications and for access to external services

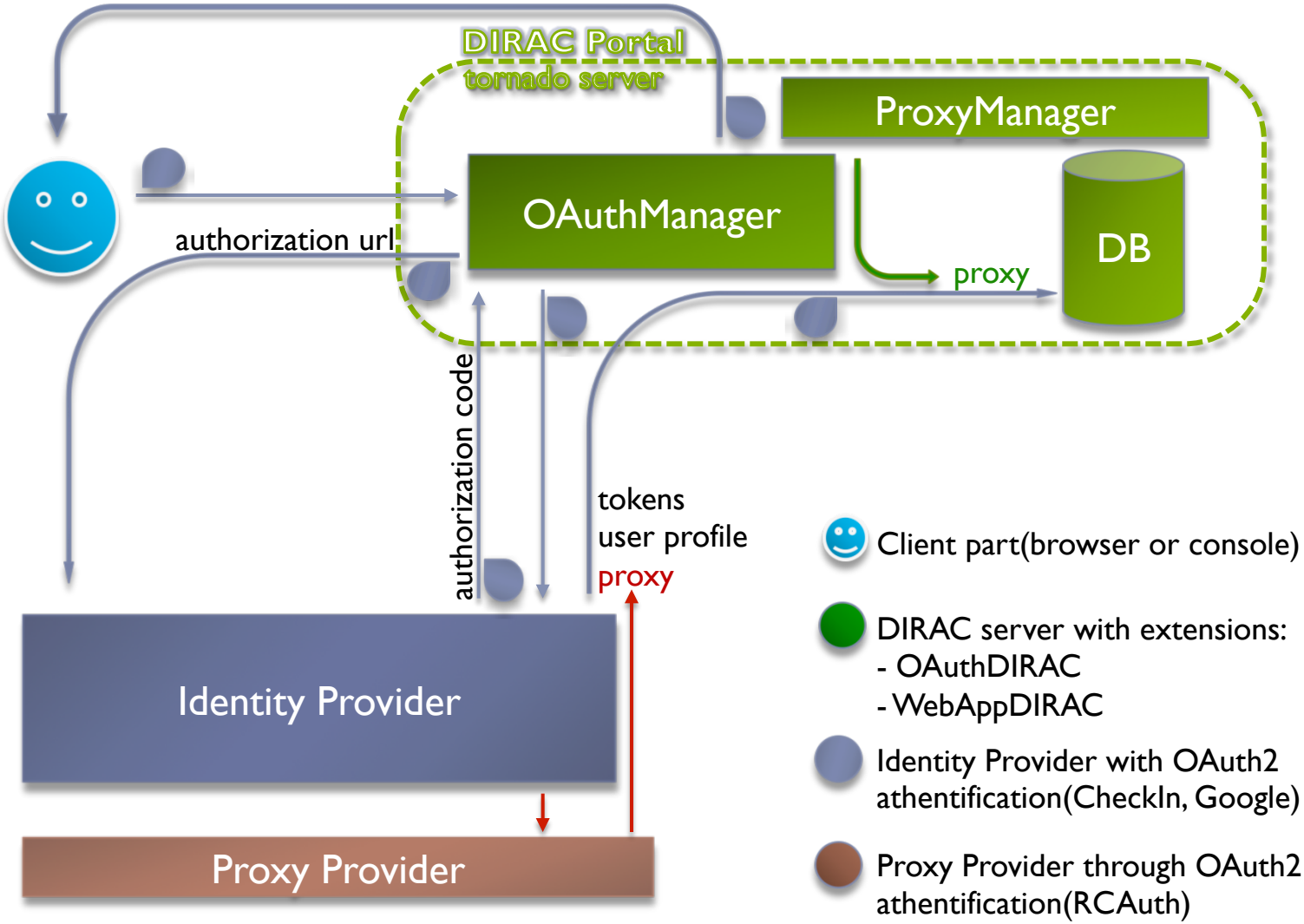
New DIRAC "SSO" components

- OAuthDIRAC extension
  - OAuthManager service + OAuthManagerClient + OAuthManagerDB
    - Generates authentication URL
    - Stores information on the user's OAuth session (session ID, AccessToken, RefreshToken)
  - AuthenticationHandler in the WebApp framework
    - Providing OAuth callback http URL
    - REST interface for the command line aithentication
- In WebAppDIRAC
  - Authentication based on OAuth token
  - User interface elements – login menu
- In DIRAC
  - CS helper utilities for new types of the configuration data

OAuth authentication flow



Authentication flow simplified



Setting up the OAuth components

- Install the DIRAC server with the extensions:
  - WebAppDIRAC
  - OAuthDIRAC
- Configure and start the OAuthManager service
- Configure and start the HTTP endpoints
  - AuthenticationHandler in the WebApp/Tornado
- Register the client in OAuth2 authentication provider, e.g. Check-In or Google
  - Set authorization flow
  - Set redirect_uri(the OAuthManager HTTP endpoint in our case)

configuration

- Set Identity Providers with some options in /Resources/IdProviders section:

```
{
 CheckIn                # Name of Identity Provider
  {
   method = oAuth2  #This option to now that provider use OAuth2 authorization protocol
   issuer = https://aai-dev.egi.eu/oidc      #This option need to get oauth2 metadata from IdP
   client_id = 2C7823B4-wqenknsadljdas2-E5D06D955809 # ID and Secret of client that you registred
   client_secret = 732h9d0dn-3_CRcUf6paEMejjojAqQz5A  #   in Identity Provider
   Syntax      # In this section we set mechanism to parse incoming information from IdP
   {
    VOMS     # In this section we decribe how to get VOMS/Role
    {
     claim = edu_person_entitlements                                  # Claim where need to search
     vo = ^urn:mace:egi.eu:(group:registry|group):<VALUE>[:#].*       # RE templete to get VO
     role = ^urn:mace:egi.eu:group:.*:role=<VALUE>[#:].*              # RE template to get Role
    }
   }
   proxy_provider = RCAuth      # Proxy Provider that able to generate proxy for user that
 }                              #   authorizated through this Identity Provider
}
```

configuration

- Set Proxy Providers with some options in /Resources/ProxyProviders section:

```
{
  RCAuth   # Name of Proxy Provider
  {
    method = oAuth2  #This option to now that provider use OAuth2 authorization protocol
    issuer = https://masterportal-pilot.aai.egi.eu/mp-oa2-server    # URL to get oauth2 metadata
    client_id = myproxy:949241khasdkhkhk358d4981d      # ID and Secret of client that you registred
    client_secret = lSh-Q32xh2pQc7rAlB_2qGVcQVNMf    #   in Identity Provider
    max_proxylifetime = 864000   # Maximum live time of proxy that Proxy Provider can create
    proxy_endpoint = https://masterportal-pilot.aai.egi.eu/mp-oa2-server/getproxy
                                                                    # URL that give access to
                                                                    #   get proxy in response
  }
}
```

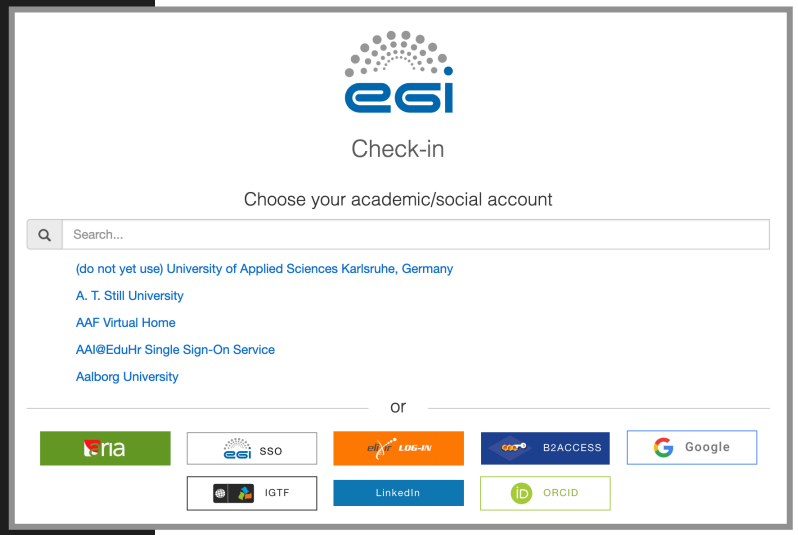
Web Portal authentication

Command line authentication

```
[[dirac@ce-emi pro]$ python DIRAC/FrameworkSystem/scripts/dirac-proxy-init.py -O CheckIn -g training_user -q
OAuth authentification from CheckIn.
Use link to authentication..
https://ce-emi.bitp.kiev.ua:9943/oauth2/oauth?getlink=MZ7XnO4iyMYTx9Vw2wkpBbHrm3Gz8f
```

```
[ Waiting 3.0 minutes when you authenticated..     ..*       $ [3~

Proxy generated:
subject      : /DC=org/DC=ugrid/O=people/O=BITP/CN=Andrey Litovchenko/CN=3461819742
issuer       : /DC=org/DC=ugrid/O=people/O=BITP/CN=Andrey Litovchenko
identity     : /DC=org/DC=ugrid/O=people/O=BITP/CN=Andrey Litovchenko
timeleft     : 23:59:59
DIRAC group  : training_user
rfc          : True
path         : /tmp/x509up_u3310
username     : alitov
```



Proxy Providers

- ProxyProvider is a new Resource type for services generating X509 certificate proxies on demand

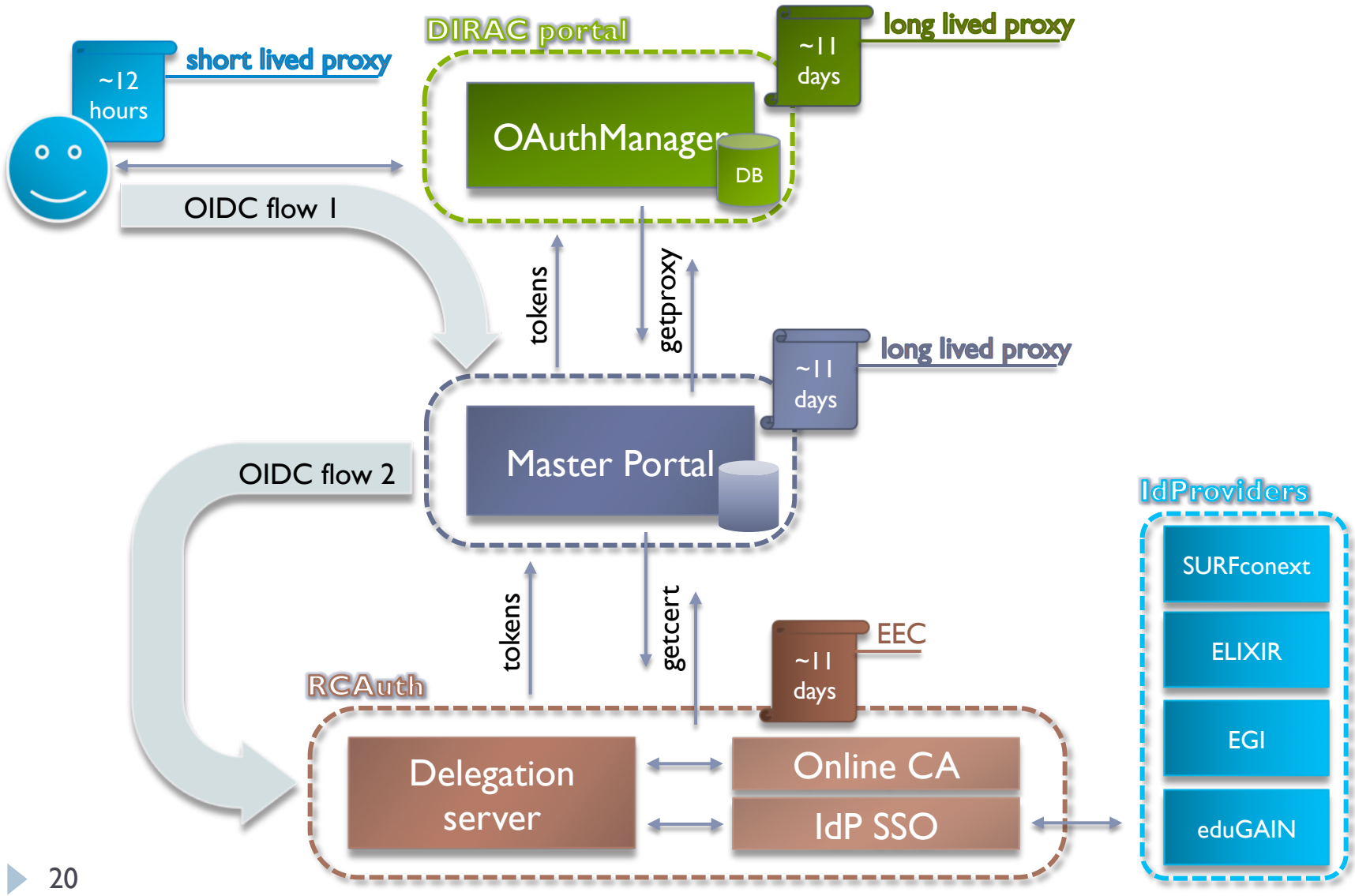
- Current implementations
  - DIRAC CA proxy provider – generates user proxy from a certificate signed by the DIRAC CA
  - PUSP proxy provider – EGI service generating user proxy out of a robot certificate with an extended DN containing user name
    - E.g. used by the fedcloud.egi.eu VO
  - RCAuth proxy provider

RCAuth

- RCauth.eu is a Research and Collaboration Authentication CA Service for Europe

- To obtain proxy certificates from the RCauth.eu online CA do not directly contact the RCauth CA, but use an intermediate service, a so-called Master Portal where you must register your client. Master Portal is an OpenID Connect Provider, with an integrated protected endpoint for obtaining proxy certificates.

RCAuth

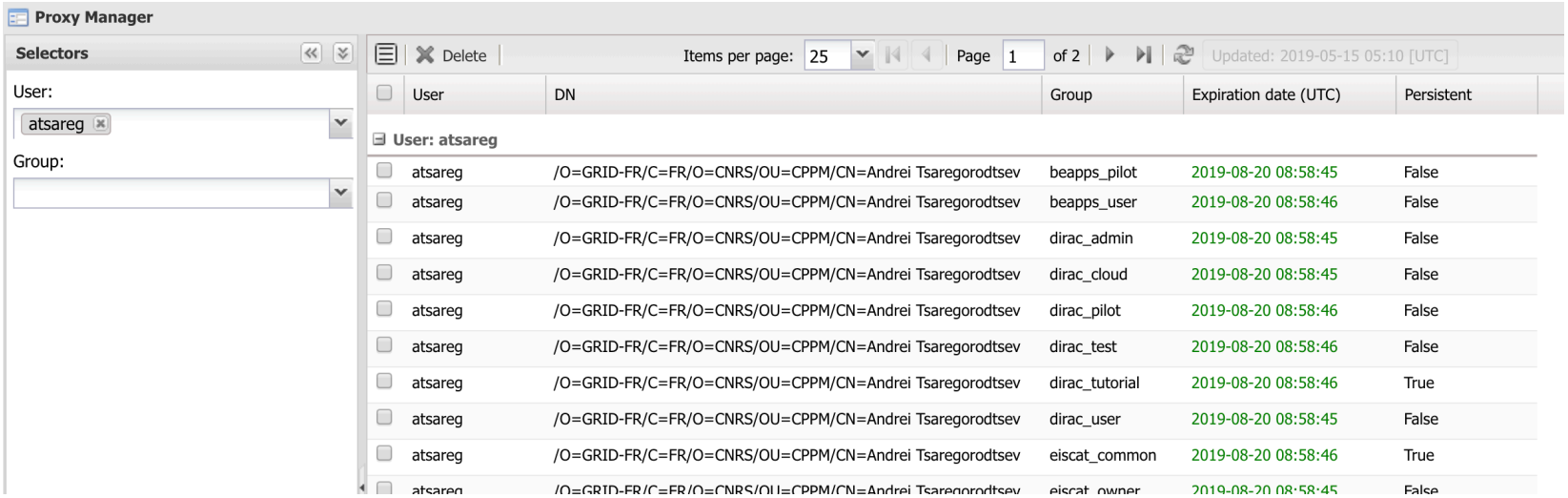

DIRAC portal
~12 hours
short lived proxy
~11 days
long lived proxy
OAuthManager
DB
OIDC flow 1
tokens
getproxy
~11 days
long lived proxy
Master Portal
OIDC flow 2
IdProviders
SURFconext
ELIXIR
EGI
eduGAIN
tokens
getcert
~11 days
EEC
RCAuth
Delegation server
Online CA
IdP SSO

Proxy renewal

- We have to have valid proxy in the ProxyManager to perform operations on behalf of the user
- With X509 certificates stored proxies are renewed once per year by the users
- Renewal of proxies provided by the DIRAC CA and PUSP service is trivial
  - Just ask for the new proxy
- Renewal of RCAuth proxy is another complex flow using the OAuth AccessToken (and most likely RefreshToken) stored in the OAuthDB
  - To be done

Proxies in the ProxyDB

- Proxies are stored in DIRAC now with embedded DIRAC group extension



User	DN	Group	Expiration date (UTC)	Persistent
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	beapps_pilot	2019-08-20 08:58:45	False
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	beapps_user	2019-08-20 08:58:46	False
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	dirac_admin	2019-08-20 08:58:45	False
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	dirac_cloud	2019-08-20 08:58:45	False
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	dirac_pilot	2019-08-20 08:58:46	False
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	dirac_test	2019-08-20 08:58:46	False
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	dirac_tutorial	2019-08-20 08:58:46	True
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	dirac_user	2019-08-20 08:58:45	False
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	eiscat_common	2019-08-20 08:58:46	True
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	eiscat_owner	2019-08-20 08:58:45	False

- Proxies returned by external proxy providers does not contain this extension
- Switching to storing only proxies without DIRAC extension
  - The extension will be added on the fly whenever the proxy delegation will be requested

DN properties

- alukkarinen
  - DN = /DC=org/DC=terena/DC=tcs/C=FI/O=CSC - Tieteen tietotekniikan keskus Oy/CN=Ari Lukkarinen
  - CA = /C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA eScience Personal CA 3
  - Email = ari.lukkarinen@csc.fi
- alytov
- alytovchenko
  - DN = /O=EGI/OU=AAI-Pilot/O=EGI Foundation/CN=Andrii Lytovchenko -hXsyo-lfs-2HPDj
  - DNProperties
    - -O_EGI-OU_AAI-Pilot-O_EGI Foundation-CN_Andrii Lytovchenko -hXsyo-lfs-2HPDj
      - ProxyProviders = CheckInProxy
      - Groups = training_user, training.egi.eu_member, training_user, training.egi.eu_owner
  - ID = 97fadf63e55ea358a4f084e4c136475e377357c6723269f23eb9aba437fd6d9d@egi.eu
  - Email = andrii.lytovchenko.workmail@gmail.com

- We should keep track of the origin of the user certificate proxy
  - To apply appropriate mechanism for the proxy renewal
  - To use the proxy only for appropriate groups
    - User can be a member of different VOs/groups using different proxy providers !

OAuth development status

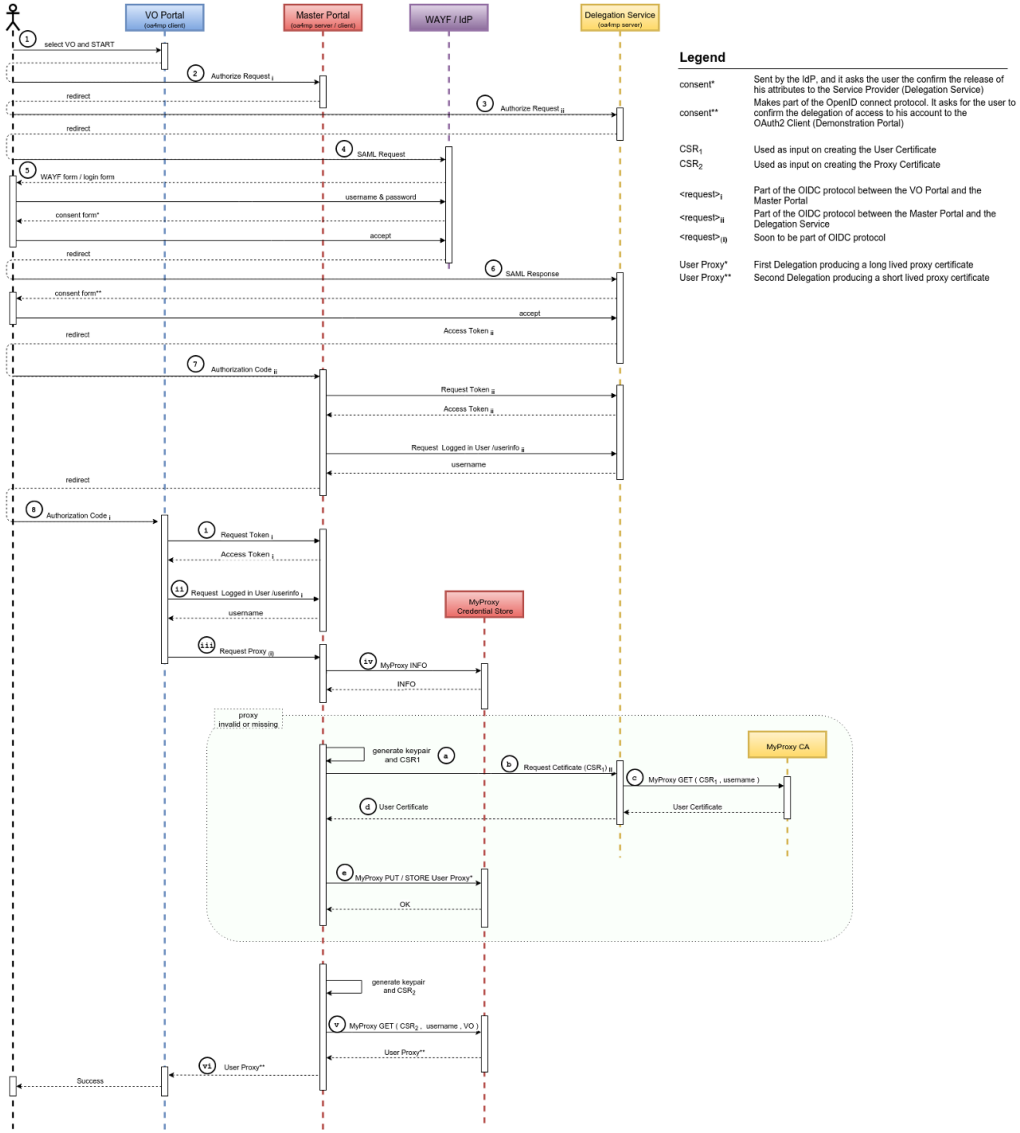
- The prototype OAuth components are demonstrated to work
  - Web Portal authentication
  - Automatic user registration
    - Interpretation of the user profile information is still to be discussed by the Check-In managers
  - On demand proxy generation with DIRAC CA and RCAuth providers
- The WebPortal authentication with Check-In is enabled in the dirac.egi.eu portal
- The code is being (re)packaged in OAuthDIRAC/DIRAC/WebAppDIRAC packages
  - Tests and docs are still to be added
- The goal is to make it available in v6r22

Conclusions

- There are many good reasons to replace the X509 based security framework by the one using OAuth/OIDC/SSO technologies
- The support of the OAuth/OIDC/SSO in DIRAC is implemented and demonstrated to work with the DIRAC4EGI service – Web Portal and command line client
- On demand X509 proxy generation is enabled with various proxy providers including the RCAuth service
- The developed software is being prepared to be available in the release v6r22

Back-up slides

RCAuth


VO Portal
(oaefmp client)
Master Portal
(oaefmp server / client)
WAYF / IdP
Delegation Service
(oaefmp server)
select VO and START
Authorize Request I
redirect
Authorize Request II
redirect
SAML Request
WAYF form / login form
username & password
consent form*
accept
redirect
SAML Response
consent form**
accept
Access Token II
redirect
Authorization Code II
Request Token III
Access Token III
Request Logged in User /userinfo II
username
redirect
Authorization Code I
Request Token I
Access Token I
Request Logged in User /userinfo I
username
Request Proxy III
MyProxy Credential Store
MyProxy INFO
INFO
proxy
invalid or missing
MyProxy CA
generate keypair
and CSR1
Request Cetificate (CSR1) III
MyProxy GET (CSR1 , username)
User Certificate
User Certificate
MyProxy PUT / STORE User Proxy*
OK
generate keypair
and CSR2
MyProxy GET (CSR2 , username , VO)
User Proxy**
User Proxy**
Success
Legend
consent* Sent by the IdP, and it asks the user the confirm the release of his attributes to the Service Provider (Delegation Service)
consent** Makes part of the OpenID connect protocol. It asks for the user to confirm the delegation of access to his account to the OAuth2 Client (Demonstration Portal)
CSR1 Used as input on creating the User Certificate
CSR2 Used as input on creating the Proxy Certificate
<requestI> Part of the OIDC protocol between the VO Portal and the Master Portal
<requestII> Part of the OIDC protocol between the Master Portal and the Delegation Service
<requestIII> Soon to be part of OIDC protocol
User Proxy* First Delegation producing a long lived proxy certificate
User Proxy** Second Delegation producing a short lived proxy certificate