An authentication solution for iRODS based on the OpenID Connect protocol

iRODS UGM 2019

Michele Carpené - m.carpen@cineca.it

iRODS UGM 2019

26-27 June 2019, Utrecht, The Netherlands

Acronyms

PAM	**Pluggable Authentication Module**. Provides dynamic authentication support for applications and services in a Linux system
Central Authentication Service (CAS)	A central endpoint which authenticates users e.g. towards a specific userbase
IdP	Identity Provider. IdPs are responsible to create/manage a user's identity and to authenticate users in a federated context.
OP	**OpenID** Provider, a specific IdP which provides authentication service using the OpenID standard
OIDC	OpenID Connect Protocol
B2ACCESS	A Central Authentication Service which acts both as an OP and as a bridge for multiple IdPs. From the B2ACCESS web page multiple IdPs can be selected by the user to authenticate

Rationale

We are going to describe an authentication solution for iRODS based on the OpenID Connect (OIDC) protocol.

More in details we are going to describe **a new Pluggable Authentication Module (PAM)**, which allows iRODS to:

- Accept an OIDC token
- Validate the token against an Authentication Service (B2ACCESS)
- Map the user to a local account using the attributes provided by the Authentication Service once validated the token.

The OIDC authentication flow

This is the typical authentication flow, where a user try to get access to a generic service exploiting the OIDC protocol:

1. The user login to a generic service
2. The service redirects the user to the OpenID Provider (OP) via OIDC protocol
3. The user authenticates himself/herself against the OP using his/her own credentials and/or the identity associated to one of his IdPs
4. The service receives a response from the OP and based on that, it allows the user to perform the required action and/or get access to the desired resource

Critical points

We want to chain together a generic front-end service and a back-end service (B2SAFE). The back-end allows users to upload/download data (e.g. from/to iRODS)

1. The user logins into the front-end service using the OIDC protocol and his/her federated identity
2. The user gets access to the B2SAFE service with the same identity, and he/she is mapped to a local account

Question: How the front-end service can pass the user identity credentials to B2SAFE to authenticate the user?

Proposed solution for the B2SAFE

We need to satisfy two basic requirements

1. The user identity credentials must be validated
2. The user federated identity must be mapped to an iRODS's local account

Solution:

1. After the user logins successfully into the front-end he/she receives a ID token, an access token and a refresh token. These tokens are released by the B2ACCESS.
2. The user tries to access the B2SAFE using the access token
3. The B2SAFE validates the token against the same B2ACCESS endpoint, and map the user to the local username (using the new PAM module)

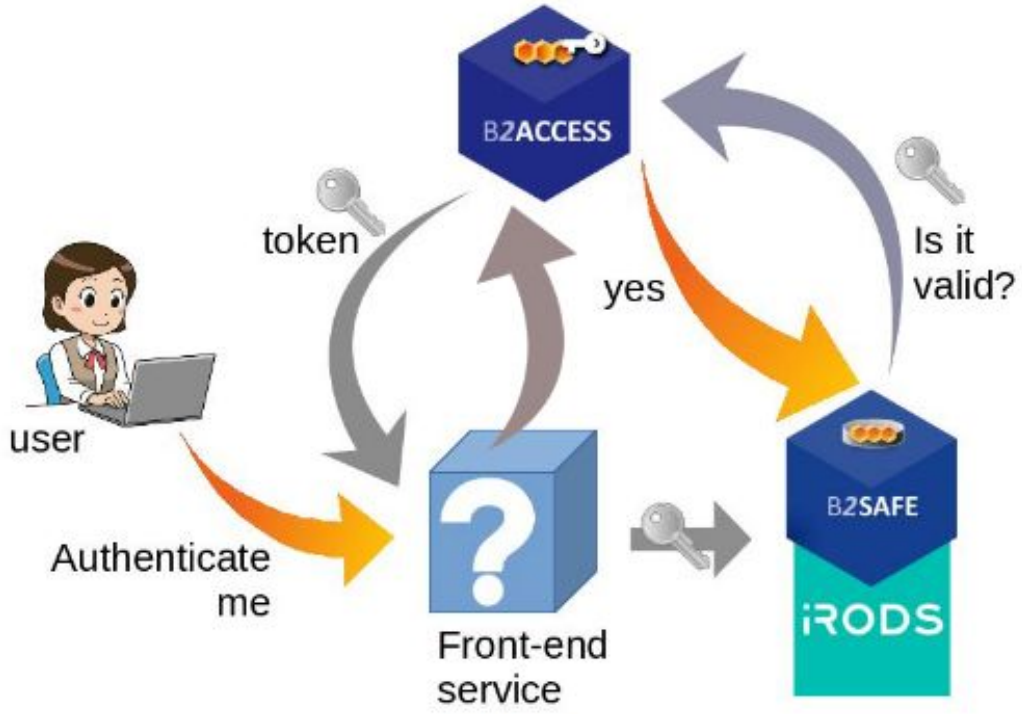

Fig. 1

Overview of the main authentication scenario

Implementation

We implemented the proposed solution as a PAM module (https://github.com/EUDAT-B2SAFE/pam-oauth2) , written in C++, which needs to be compiled and configured.

Using PAM as a framework permits to have more flexibility compared to other solutions (e.g. new iRODS authentication plugin)

How it works

Using the iRODS PAM mechanism the user has to login with the PAM authentication method, but instead of the password of the iRODS local account he/she uses the access token.

The PAM module **pam_oauth2.so** receives the token and issues a request to the B2ACCESS's **token_validation_ep**

{

“email”: ”roberto@email.com”,

“Token_type”: “Bearer”,

“exp”: 1520001942,

“iat”: “1519998342”,

}

Two Concrete Use Case

The solution has been tested with two front-end services in the context of the EOSC-hub project (https://www.eosc-hub.eu)

1. The first one is an HTTP interface (https://github.com/EUDAT-B2STAGE/http-api), which ex-poses some functions to upload/download data using the iRODS python library (https://github.com/irods/python-irodsclient)
2. The second one is a data management tool called DataHub (https://www.eosc-hub.eu/services/EGI%20DataHub), which is able to mount external storage if this storage is exposed through a WebDAV interface.

In the second use case the DataHub takes care both of user authentication and hence of token refreshing.

Benefits and Limits

Proposed solution solves the aforementioned authentication issue and reuse the OIDC tokens without requiring the user to login twice.

1. Flexibility in the account mapping
2. Possibility to create a user on the fly

Limits:

1. We need to know the local iRODS username at the login time, because this is required by iRODS PAM
2. The token has limited lifetime (refresh token could be used, but front-end must support it)

Conclusions

We have described the implementation of a solution to add support of the OpenID Connect protocol to iRODS relying on its PAM authentication mechanism

1. Flexible solution for different use cases
2. Single identity based
3. Validation of OIDC credentials

Future developments can be envisioned about user mapping… (dynamic approach vs static approach, regular expressions...)

Acknowledgements

The implementation of the PAM module for the OIDC protocol started forking the code of the OAuth2 PAM module by Alexander Kukushkin (https://github.com/CyberDemOn/pam-oauth2)