Contribution ID: **74** Type: **Presentation**

Supporting Keycloak in iRODS systems with OpenID authentication

Monday, 27 January 2020 15:57 (6 minutes)

The Integrated Rule-Oriented Data System (iRODS) is an open source data management software aimed at deployment in mission critical environments,
which virtualizes data storage resources. iRODS can take advantage of OpenID login by using an authentication plugin.
The approach works by letting iRODS authenticate using tokens provided by an OpenID provider, which are verified by the plugin (e.g. [1]).

However, the token is transferred between the iRODS client and the iRODS server by using the username field,
which has a maximum length of 1024+64 bytes. This is plenty for most OpenID implentations;
the only exception the authors are aware of is Keycloak, since its JWT tokens contain extensive information about the user,
their identity and their permissions. Some changes were required in the different components of the authentification plugin
to avoid issues with such tokens.
Additionally, when such a token is sent from the iRODS client to the iRODS server, an iRODS USER_PACKSTRUCT_INPUT_ERR error
is produced, and the iRODS server prints a URL to enable re-authentification.
Once this URL is used by the user to authentificate, the iRODS command can complete.

While this is enough in some situations, some workflows cannot use this approach.
In particular, the iinit command will not work, and web-based applications will require modifications.
There are plans within iRODS to more tightly integrate OpenID in the future, avoiding these issues and simplifying implantation.
However, this solution will very probably be provided outside of our LEXIS project timeframe (2019-2021).

We implemented two solutions to the issue, taking care to maintain the overall security level:

a) for web-based applications interfacing directly with the user, a helper thread or process is launched to perform a query for (meta-data)
to the iRODS system and later (after successful authentication) store the result. The main thread gets the authentication URL from the helper
thread (while the helper is waiting for results from the iRODS) and redirects the user to it. The helper thread obtains the result data from
the iRODS server as soon as the user authenticated. After the user is eventually redirected to the web portal, the web application can retrieve
this data. No token ever traverses the iRODS client-server interface, so no error is produced.

b) for back-end applications, the solution above is not applicable. We therefore modified the iRODS authentication microservice [2] to implement
opaque tokens. The microservice now sends a hash of the token to the user (which is small enough to traverse the iRODS interface). Upon receiving
a hash, the microservice database is used to retrieve the token and perform the verification activities.

The project received funding from the European Union's Horizon 2020
research and innovation program under grant agreement no. 825532 with

LEXIS - Large-scale EXecution for Industry & Society.
We thank the ITS Group of Leibniz Supercomputing Centre (LRZ, Garching) of the BAdW for supporting our research with its Compute-Cloud infrastructure.

[1] https://iRODS.org/uploads/2019/Cacciari-CINECA-OpenID_Connect-paper.pdf
[2] https://github.com/heliumdatacommons/auth_microservice/

Primary authors: Dr GARCÍA-HERNÁNDEZ, Rubén Jesús (Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities); Mr GOLASOWSKI, Martin (VSB Technical University Ostrava)

Presenter: Dr GARCÍA-HERNÁNDEZ, Rubén Jesús (Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities)

Session Classification: Site reports

Track Classification: CS3 Community Site Reports