

OIDC support for SSH

Diana Gudu, Marcus Hardt, Gabriel Zachmann, Jonas Schmitt
Karlsruhe Institute of Technology

hardt@kit.edu



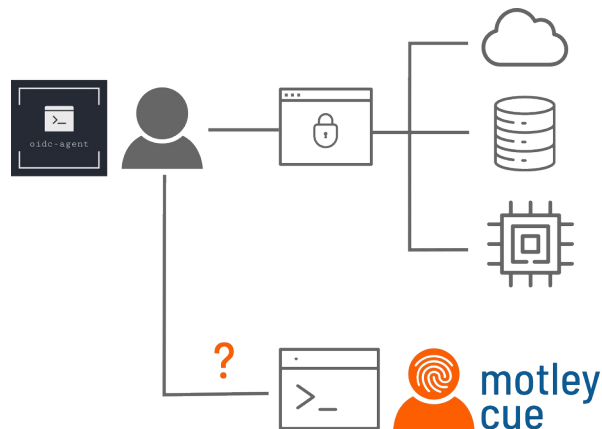
Motivation

- Enable federated access to shell-based services
 - Federated Identity Management → OpenID Connect (**OIDC**)
 - Shell-based services → Secure Shell (**SSH**), local identities



Our solution: server & client side tools

- Work with standard SSH software
- Use OIDC tokens for AuthN & AuthZ
- Manage local identities





Why would you use it?

...as a user

- Single Sign-On (SSO)
- No additional service credentials
- No need for SSH key management
- No prior registration



Why would you use it?

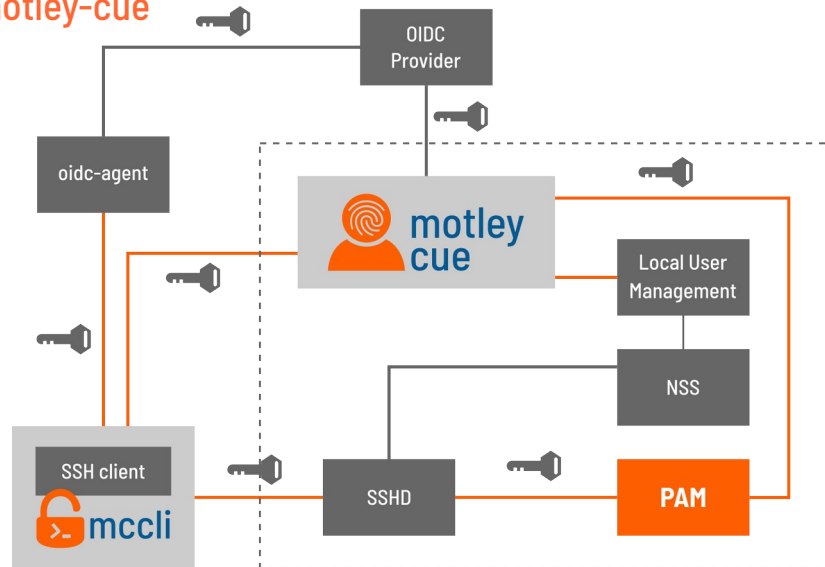
...as a service provider

- Benefits of federated AAI
 - Offload identity management to home organisation
 - Offload authorisation management to federation (VOs)
- Bridges the gap from federated to local identity
 - Manages the mapping of federated to local accounts
 - Manages the lifecycle of local accounts (create, update, suspend)
 - OIDC-based authentication → no need for managing additional credentials (passwords, ssh keys)
 - Manages access control based on federated authorisation models

Approach

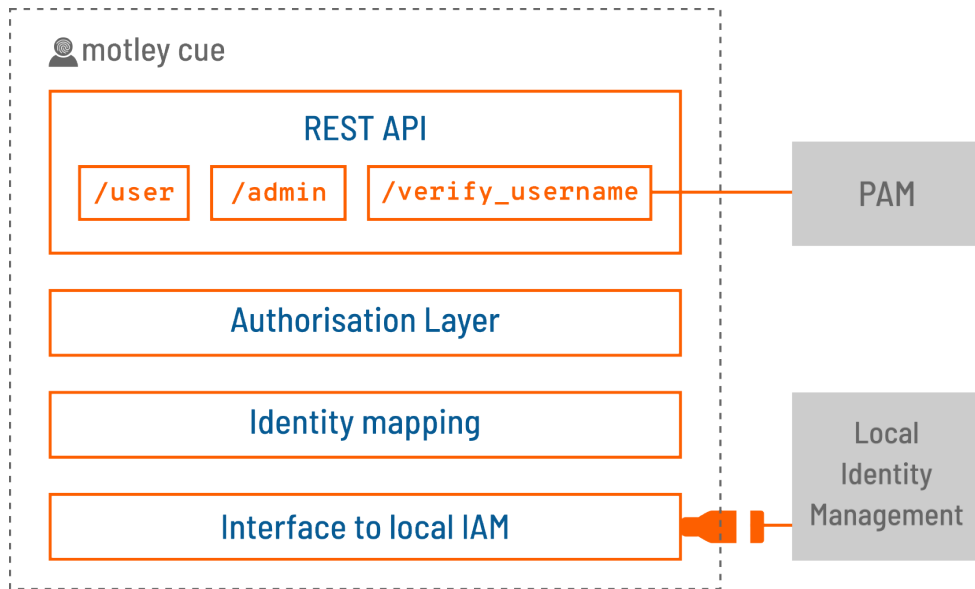
- Server side:
 - Use PAM module with oidc support: **pam-ssh-oidc** (PSNC/Pracelab.pl)
 - Add REST interface to ssh-server to manage the details: **motley-cue**
- Client side:
 - oidc-agent** for obtaining tokens
 - Enable **ssh-clients** to use tokens

No modifications of **ssh** or **sshd**



Server Side

motley-cue architecture





Authorisation

- Support for multiple OIDC Providers
- Based on VO membership
- Based on assurance
- Individual users via sub+iss



Local User Management

- Interface to site-local identity management systems
 - Extensible, plug-in architecture
 - Supported identity backends: UNIX accounts, LDAP, KIT RegApp
- Identity mapping: **sub + iss → local username**
 - Stored directly in the local IdM system
 - username generation strategies → uniqueness
 - Friendly: preferred username, first_last, ...
 - Pooled: egi001, egi002, ...
 - VOs mapped to local groups



Advanced features

- Approval workflow → admins oversee all deployment requests
- LDAP backend → for managing local accounts
- Audience → restrict access to tokens released for configured audience
- Long tokens → 1kB too long for SSH, generate one-time tokens



Nice to know

- SSH daemon is not modified
- PAM module may be combined with other modules
 - Possible: ssh-key + password + OIDC + 2nd factor (linotp)

Client Side

SSH Clients

- 2 Simple changes on the command line:
 - add our wrapper tool mccli
 - replace username with identity provider

Old: `ssh marcus@ssh-oidc-demo.data.kit.edu`

New: `mccli ssh ssh-oidc-demo.data.kit.edu --oidc egi`

- Tools to install:

```
$ pip install mccli
```

```
$ apt-get install oidc-agent
```

- Again: packages provided for all major Operating Systems

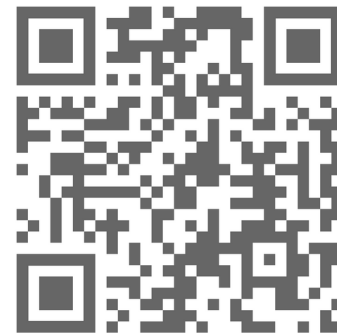


SSH Clients



- Everything is different on Windows ;)
- Putty SSH Client required source code modifications
 - Joint effort with Simon Tatham (putty main developer)
 - General Plugin Interface (available in putty-0.78:
<https://www.chiark.greenend.org.uk/~sgtatham/putty/prerel.html>)
- Plugin and oidc-agent installed and shipped together
<http://repo.data.kit.edu/windows/oidc-agent>





Recorded Demo

- This demo shows the first-time setup on windows
- Choices are cached. User only enters password **once** (for each windows reboot)

A screenshot of a Windows terminal window titled "Index of /windows/oidc-agent/" and "Success". The terminal shows the following output:

```
Success: Enter short name for the account to configure: egi
Generating account configuration ...
accepted
Success: To continue and approve the registered client visit the following URL in a Browser of your choice:
https://lsl.egi.eu/auth/realms/egi/protocol/openid-connect/auth?response_type=code&client_id=oidc-agent&redirect_uri=https://localhost:8888&scope=openid%20profile%20email%20offline_access%20udperson_entitlement%20udperson_scoped_affiliation%20udperson_unique_id&prompt=consent&state=0:02xE8PABGFHbJULJ55MEL5Qyp:57:QzovW0llcnMvVXNlcj08CHBEYXRhL0xvY2Fs1R1RlDkAVY0211KYi0MT1EVKsvb2lKYy1hZ2VudC4x&access_type=offline&code_challenge_method=S256&code_challenge=FVYBaYt0Gv0cAvuiesYob1F0gk89psu_Q3z=51Q
Please click on the URL above to continue.
oidc-agent: Polling oidc-agent to get the generated account configuration .....success
The generated account config was successfully added to oidc-agent. You don't have to run oidc-add.
Enter encryption password for account configuration 'egi': Confirm encryption Password: _
```

The terminal window is overlaid on a browser window showing a "Success" message. The Windows taskbar at the bottom shows the system tray with the date 9/14/2022 and time 10:16 AM.



Live Demos

- 1. Usage from linux



- 2. Usage from windows



Live Demo Server

for **You** to try this



<https://ssh-oidc-demo.data.kit.edu/>

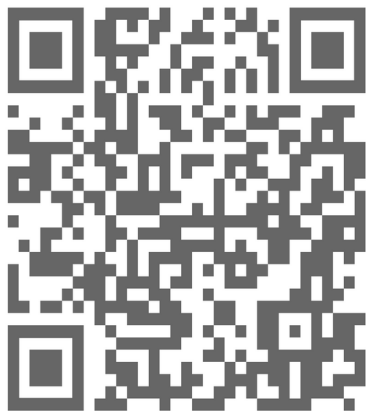
Contributors

- PAM module (pam-ssh-oidc): Pracelab.PL (Pawel Wolniewicz (PSNC), Damian Kaliszan (PSNC))
- User provisioning (feudal): KIT (Lukas Burgey, Joshua Bachmeier, Diana Gudu, Marcus Hardt)
- Integration serverside (motley_cue): HIFIS (Diana Gudu (KIT), Andreas Klotz (HZB))
- HPC Integration and testing: EOSC-Synergy (Diana Gudu (KIT), Rubén Díez, CESGA))
- Integration, consulting, and review: Enol Fernandez (EGI), Viet Tran (IISAS), Mario David (LIP), Mischa Salle (Nikhef)
- Infrastructure Manager Integration: Miguel Cabeller (UPV), German Molto(UPV)
- oidc-agent integration: KIT (Gabriel Zachmann (KIT))
- putty-integration: Dmytro Dehtyarov (KIT/GEANT), Jonas Schmitt (KIT), Simon Tatham (Putty), Niels van Dijk (SURFnet)



More information

- Download oidc-agent for Windows & PuTTY



<https://repo.data.kit.edu/windows/oidc-agent>

- Demo server



<https://ssh-oidc-demo.data.kit.edu/>

- Contact



m-contact@lists.kit.edu