motley-cue: SSH access with OIDC tokens

Diana Gudu, Marcus Hardt, Gabriel Zachmann
Karlsruhe Institute of Technlogy

gudu@kit.edu

Motivation

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

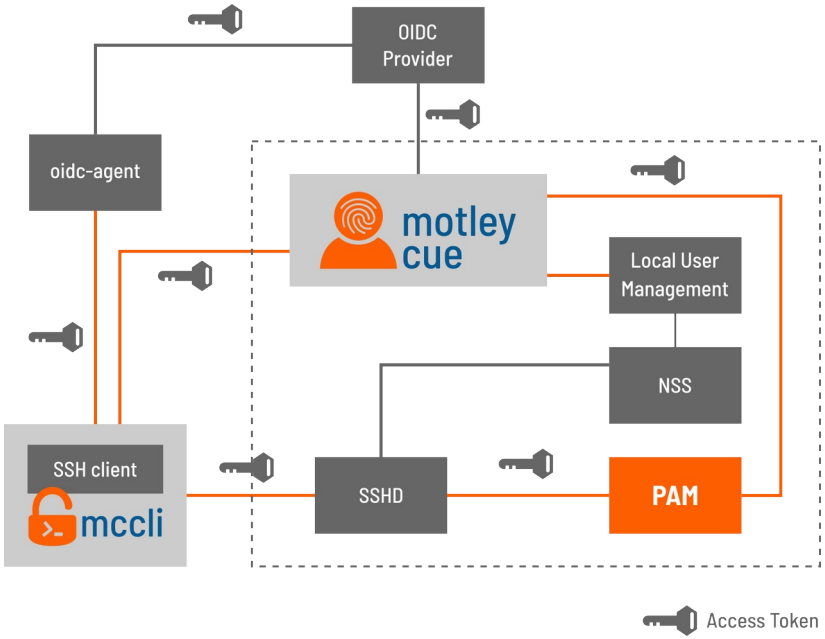
Motivation

- **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

- Works with standard SSH software
- Uses OIDC tokens for AuthN & AuthZ
- Manages local identities

How does it work?



- Client side
  - mccli → wrapper for SSH client
  - oidc-agent → manage OIDC tokens
- Server-side (*this talk*)
  - **motley-cue** → authorisation & account provisioning
  - PAM[1] → token prompt & validate @motley-cue (AuthN)
  - Unmodified SSHD → keyboard interactive

[1] developed at PSNC

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

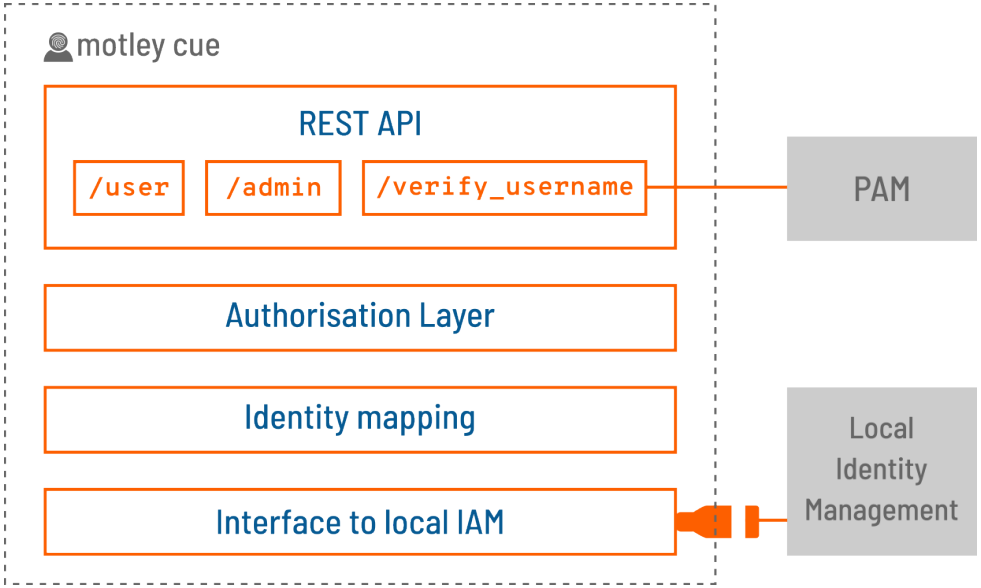
- Benefits of federated AAI
  - Offload identity management to home organisation
  - Offload authorisation management to federation (VOs)

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

motley-cue architecture



Authorisation

- support for multiple OIDC Providers
- based on VO membership
- 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

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

New 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

Technical details

- Easy deployment

Technical details

- Easy deployment
  - Packages for most common Linux distributions

http://repo.data.kit.edu

Technical details

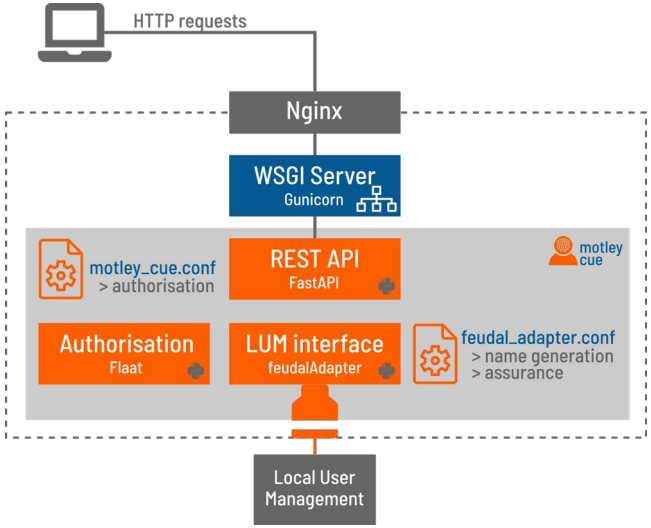
- Easy deployment
  - Packages for most common Linux distributions
  - systemd integration

```
$ apt install motley-cue pam-ssh-oidc
$ vim /etc/motley_cue/motley_cue.conf
$ systemctl restart motley-cue
```

http://repo.data.kit.edu

Technical details

- Easy deployment
  - Packages for most common Linux distributions
  - systemd integration
- Python, FastAPI




```
$ apt install motley-cue pam-ssh-oidc
$ vim /etc/motley_cue/motley_cue.conf
$ systemctl restart motley-cue
```

debian openSUSE

http://repo.data.kit.edu

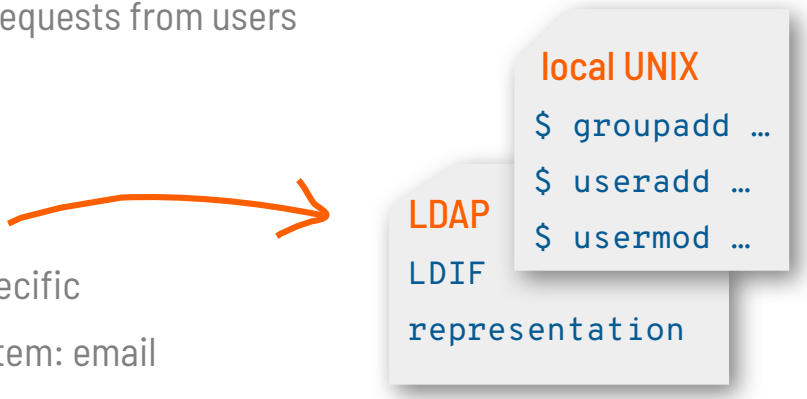
Demo

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

Approval workflow

https://github.com/dianagudu/egi-2022-demo

- Admins can oversee all deployment requests from users
- How it works:
  - User triggers **deployment**
  - Admin (and user) is **notified**
    - notification is backend-specific
    - supported notification system: email
  - Admin **accepts** or **rejects** the request manually
  - Users are *not* notified of acceptance/rejection → pull model
- Subsequent deployment requests
  - notify the admin only when updates are necessary



LDAP backend

https://github.com/dianagudu/egi-2022-demo

- Local accounts are managed in an LDAP
  - OIDC unique ID stored in a configurable attribute
  - Required LDAP schemas: inetOrgPerson, posixAccount, posixGroup
- Modes
  - **read-only**: local user management fully controlled by LDAP admins, including mapping
  - **pre-created**: motley-cue adds the mapping information to pre-created accounts
  - **full-access**: motley-cue has full control to provision users and groups in LDAP

Future work

- Account **deprovisioning**
- More flexible **VO → local group** mapping
  - regex filtering and naming
- **mytoken** integration

Future work

- Account **deprovisioning**
- More flexible **VO → local group** mapping
  - regex filtering and naming
- **mytoken** integration

- Increase **adoption**
- Currently being evaluated by
  - Helmholtz Cloud → cloud orchestration for imaging use case
  - PUNCH4NFDI → compute resources for particle physics

More information

- Demo instance

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

- Documentation

https://github.com/EOSC-synergy/ssh-oidc

https://motley-cue.readthedocs.io

- Contact

m-contact@lists.kit.edu