SSH access with OIDC tokens

Diana Gudu, Marcus Hardt
Karlsruhe Institute of Technology

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

Our solution: server & client side tools
- Works with standard SSH software
- Uses OIDC tokens for AuthN & AuthZ
- Manages 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)
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)
  - Manages access control based on federated authorisation models
  - OIDC-based authentication → no need for managing additional credentials (passwords, ssh keys)
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

1^developed at PSNC
Approach

- **Server side:**
  - Use PAM module with oidc support: `pam-ssh-oidc` (PSNC/Pracelab.pl)\(^1\)
  - 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

\(^1\)developed at PSNC
Server Side
motley-cue architecture

REST API

/user
/admin
/verify_username

Authorisation Layer

Identity mapping

Interface to local IAM

PAM

Local Identity Management
Authorisation

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

- Interface to site-local identity management systems
  - Extensible, plug-in architecture
  - Supported identity backends: UNIX accounts, LDAP, KIT RegApp
Account provisioning

- 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
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
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
Technical details

- Easy deployment
  - Packages for most common Linux distributions
  - systemd integration
- Python, FastAPI
- Nice to know
  - SSH daemon is not modified
  - PAM module may be combined with other modules

Possible:
ssh-key + password + OIDC + 2nd factor (linotp)

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

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

  Old:       ssh diana@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
Demo

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

https://youtu.be/IrwnnHmxbTQ
Demo Windows

- Short version of demo: assume oidc-agent already set up
- Choices are cached. User only enters password once (for each windows reboot)
Additional requirements

✔ Mitigate sharing of SSH keys → by not using SSH keys, but access tokens for AuthN
✔ Non-interactive client logins → with oidc-agent integration
✔ Delegation → via oidc-agent forwarding, the token is available on server
✔ MFA → possible with additional PAM modules
✔ Revocation → two options:
  ▪ Revocation of tokens (access token / refresh token) possible
  ▪ /admin endpoint to suspend/resume users
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)
More information

- Download oidc-agent for Windows & PuTTY
  - [Download link](https://repo.data.kit.edu/windows/oidc-agent)

- Documentation
  - [GitHub repository](https://github.com/EOSC-synergy/ssh-oidc)

- Contact
  - m-contact@lists.kit.edu
Demo Windows

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

- We support a long list of OPs
  - Helmholtz AAI, EGI Check-in, DEEP IAM, WLCG, Google, ...
- Only require a valid AT (JWT or not e.g. Google)
- AuthZ based on: VO membership, assurance, user whitelisting
  - federated identity should support this
  - AT should contain required scopes
    - typically: groups/eduperson_entitlement, eduperson_assurance
- Audience only if supported
Client requirements

- Supported platforms: Linux, MacOS, Windows

- Requirements Linux & MacOS:
  - python + oidc-agent + mccli
  - unmodified SSH client
  - can also use bare SSH for subsequent logins with user interaction if
    - local account deployed & known, and
    - AT can be obtained from other sources

- Requirements Windows:
  - PuTTY v0.78 (currently pre-release)
  - oidc-agent (also installs oidc-plugin for PuTTY)
Server requirements

- Supported platforms: Linux
- Additional software:
  - motley-cue
  - nginx
  - pam-ssh-oidc
- Unmodified ssh server
- Configuration:
  - sshd with challenge response authentication
  - Custom PAM module
  - Open port for HTTP requests
PAM-OIDC

- Based on OIDC access token authentication
  - user is prompted for an **Access Token** instead of Password
- Written in **C**
- Query **motley_cue** service API for:
  - token validation
  - authorisation
  - username match

```
$ curl -X 'GET' \
  $motley_cue_endpoint/verify_user&username=$username \n  -H "Authorization: Bearer $token"
{
  "state": "deployed",
  "verified": true
}
```
Approval workflow

- 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

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

- 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

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