Karlsruhe Institute of Technology

Evolution of SSH with OpenID Connect

Lukas Brocke, Diana Gudu, Marcus Hardt, Gabriel Zachmann

Mar 2024

Motivation

Motivation

- Use benefits of federated identity management with **`ssh`**
  <=> Enable using **`ssh`** with **`eduGAIN`** login
- Reduce need for **`ssh-keys`**
  - Often not encrypted
  - Sometimes shared
  - Cumbersome provisioning processes (upload via web)
- Improve life of operators
  - Scaling is hard: key approval and distribution
  - User offboarding difficult (key binding is permanent)
    - ssh-keys live forever
- Fix ssh annoyances:
  - SSH-keys trusted permanently
  - SSH-keys can be shared across devices or teams
  - SSH-key passphrase cannot be enforced

Recap: Federated Identities

- Home Organisation (university, institute)
  - Authoritative source of
    - Identifier, Name, Contact Information
    - Affiliation with the institute
    - Basic Entitlements
    - Identity Assurance
- Community AAI
  - Delegated authorisation:
    - Community Group Membership (Entitlements)
    - Direct community authorisation (Entitlements)
  - Better defined set of attributes

Federated Identities [2/2]

- As a user
  - Single Sign-On (SSO)
  - No additional service credentials
    => Increase in security
  - Less prior registration
    => Increase in convenience
- As a service
  - Offload identity management to home organisation
  - Offload authorisation management to Community AAI (VOs)
    => Reduce cost
    => Improve user data quality
    => Service is provided to a VO (not to an individual user anymore)
  - Improved security
    - Well defined incident response
    - Federated Security Operations

ssh vs federated-IDs

Feature	ssh / local-ID	eduGAIN / federated ID
Single Sign On	no	yes
Home-Org Identity Management	no	yes
Unix Shell Access	yes	no
<=> HPC Access	yes	no
git, rsync	yes	no
Shared credentials	yes	no
Permanent credential lifetime	yes	no
Unencrypted ssh-keys	yes	undefined
Federated (global) Authorisation (via VOs)	no	yes
Revocation / Offboarding	no	yes
Different credential per server	yes	no

One step back

Map federated IDs to Unix accounts

Why on earth?

1. We already have information about the federated user
    - Federated ID is conveyed via the OIDC Access Token
2. We already have authorisation information in the federated information

- The world consists of thousands of Unix servers

  => We do not want to memorise different Unix accounts for each server (hundreds!!)
    - Nobody wants to memorise all those accounts

How do we map federated ID to local Unix account

- **`motley-cue`**: https://motley-cue.readthedocs.io
  - Server-side REST interface developed in HIFIS (Germany)

- Check authorisation
  - Based on **`entitlement`** (i.e. VO) + **`assurance`** + **`sub@iss`** (user whitelist)
- If authorised: return mapped Unix account
  - If mapping exists
    - Fine, we're done
  - If mapping does not exist (Optional)
    - Dynamically provision a user
      - Pooled-accounts, "Friendly" username, External username lookup
    - Multiple provisioning backends
      - Local, LDAP, Ticket-System
- Admin interface for security incidents
  - Suspend / Resume user

Cornerstones for KIT SSH-OIDC

- Use federated identity
  - SSH-Server has no direct relation with Organisation where user comes from
- Use federated authorisation
- Support for revocation (deprovisioning / offboarding)

- Create an ecosystem of components for **`ssh`**
  - Easier to support different use-cases
- Don’t modify **`ssh`**
  - Patching **`ssh`** would result in **`forking`** => No

Implementation

Make **`ssh`** use Access Tokens

- PAM (Pluggable Authentication Module)
  - Standard Unix interface
  - Simple + well understood
  - Change the user prompt of ssh: **`Password`** -> **`Access Token`**
  - Verify Access Token
- Packages available for most Linux distributions
  - **`pam-ssh-oidc`** + **`pam-ssh-oidc-autoconfig`**
  - **`debian`**, **`ubuntu`**, **`centos`**, **`alma`**, **`fedora`**, **`rocky`**, **`SuSE`**, **`Arch`**
- User sends Access Token instead of password:

```
$ ssh testuser@ssh-oidc-demo.data.kit.edu
(tetsuser@ssh-oidc-demo.data.kit.edu) Access Token:
```

Get Access Tokens

- **`oidc-agent`**: https://indigo-dc.gitbook.io/oidc-agent
  - Just like **`ssh-agent`**
- Pros:
  - Secure
  - Easy to use
  - Largely non-interactive
- Cons:
  - Requires installation on (own) ssh-client computer

Map fed. User to Unix account

- **`motley-cue`**:small>https://motley-cue.readthedocs.io
- Client calls **`motley-cue`** before ssh

  => get the unix **`username`**
- Only issue:
  - Opens a REST interface (on a server close to **`sshd`**)

Map federated user: client side

New tool: **`mccli`**

- Automation of client-side tasks:
  - Obtain Access Token via **`oidc-agent`**
  - Get username for ssh server
    - via **`motley-cue`** REST interface
  - Put Access Token or **`OTP`** into password field

```
# Example
$ mccli ssh ssh-oidc-demo.data.kit.edu --oidc kit.edu

testme@ssh-oidc-demo:~$
```

- Pros:
  - PoC done: Federated **`ssh`** works!!
  - Many features integrated
- Cons:
  - Limited set of commands
  - Client software installation required

Usability Optimisation [1/2]

- Run client side in **`web browser`**
  - Get the Access Token
  - Fix software installation requirements
    - Ported **`mccli`** to **`javascript`**
- Simple web login (authorization-code-flow)
- Result: web-shell in your browser
- Example: https://ssh-oidc-web.vm.fedcloud.eu

What is still missing

- Advanced usage of **`ssh`**:
  - Tools on top of **`ssh`**: **`rsync`**, **`git`**
  - Advanced commandline: **`dd if=/dev/sda | pbzip2 | ssh fileserver "cat - > backup.img.bz2"`**
- Reduce dependency on client-side components
  - **`mccli`**: Find a way to drop it
  - **`oidc-agent`**: if needed once per month, a web **`copy+paste`** flow is viable.

ssh-certificates

Recap: `ssh-certificates`

- Introduced with **`openssh-v5.3`** in 2010
- ssh-certificates are not X.509
- ssh-ca signs: **`host`** keys + **`user`** keys
- ssh-certificates expire :)
- SSH server admin trusts CA to issue certificates only to trusted users!
- Different usage:
  - Provision public keys -> Provision **`ssh-ca`** keys

ssh-cer

- Certificates are ssh-k
- Certificates allow sim
  - **principals**, => lis
  - **force-command**, =>

```
$ ssh-keygen -L -f user-key-cert.pub
user-key-cert.pub:
    Type: ssh-ed25519-cert-v01@openssh.com user certificate
    Public key: ED25519-CERT SHA256:rkSKv...
    Signing CA: ED25519 SHA256:xw9aV... (using ssh-ed25519)
    Key ID: "whatever"
    Serial: 0
    Valid: from 2023-09-01T14:30:00 to 2023-09-02T14:30:00
    Principals:
        oinit
    Critical Options:
        force-command oinit-switch marcus
    Extensions:
        permit-X11-forwarding
        permit-agent-forwarding
        permit-port-forwarding
        permit-pty
        permit-user-rc
```

oinit

New components to integrate ssh-certificates with **motley-cue**

- **oinit-ca**, an online ssh-ca
  - With REST interface (protected with OIDC)
  - Authorisation (via **motley-cue**):
    1. **CA**: Authorisation based on OIDC claims
    2. **motley-cue**: Find unix username for federated user
    3. **sshd**: Ensure a local user exist for federated ID
  - Optional Provisioning
    - **oinit-ca** supports provisioning via **motley-cue**
- **oinit openssh**: serverside components to support **oinit**
  - Mapping of federated ID -> local unix account

`oinit` client side

- **`oinit`**:
  - Helper tool to configure **`ssh`** to use **`oinit`** for selected ssh servers
  - Define which ssh-servers support **`oinit`**
    - ⋯ and which **`oinit-ca`** to use
    - Once per computer lifetime
  - **`oinit add <ssh-server>[:<port>] http[s]://<ssh-ca>[:<port>]`**
- Purpose of configuration:
  - Obtain **`ssh-certificate`** whenever needed
    - e.g. expired, deleted, ⋯
  - **`oinit`** mechanism is then used via **`~/.ssh/config`**

Actual **`ssh`** call

- On first call to **`oinit`**-enables ssh-server, **`openssh-client`** will
  - Prompt user for **`oidc`** credentials
    - e.g. via **`oidc-agent`** or **`mytoken`**
  - Store ssh-certificate in **`ssh-agent`**
  - Refreshed once per **`ssh-certificate`** lifetime:
  - And do ssh:

```
$ ssh ssh-server.edu

[1] https://aai-dev.egi.eu/auth/realms/egi
[2] https://aai.egi.eu/auth/realms/egi (Accounts: egi)
[3] https://accounts.google.com
[4] https://iam.deep-hybrid-datacloud.eu
[5] https://login-dev.helmholtz.de/oauth2
[6] https://login.helmholtz.de/oauth2
[7] https://oidc.scc.kit.edu/auth/realms/kit
[8] https://wlcg.cloud.cnaf.infn.it
? Please select a provider to use [1-8]: 2
✔ Received a certificate which is valid until 2024-03-08
14:04:20
```

Future work

Future work

- Fix packaging
  - Not all tools interoperate out of the box just yet
  - **`debian`**, **`ubuntu`**, **`centos`**, **`alma`**, **`fedora`**, **`rocky`**, **`SuSE`**, **`Arch`**
- Provide consistent documentation of our Ecosystem
- Further integration
  - e.g. with Account LInking SErvicE (ALISE)
- Security Audit
- **`oidc-agent`** forwarding
- Policy: More entries on the “How-long-should-an-**`xxx`**-live?” list:
  - X.509
  - Access Token
  - Refresh Token
  - mytoken
  - ssh certificate

Summary

- No **`ssh`**-daemons (or clients) were hurt in this project:
  - Unmodified SSH Client and Server
  - Backward compatible with: password, ssh-keys, 2nd factor modules, …
- Supported platforms
  - Windows: Putty
  - Mac/Linux: OpenSSH
- Packages: https://repo.data.kit.edu
- Video: https://youtu.be/090D4s0TNaA
- Visit https://ssh-oidc-demo.data.kit.edu to try it yourself

More SSH Approaches

- Multiple different approaches exist
- Smart Shell
  - AWI, SURF
- SSH Certificates
  - DEIC
- PAM Module
  - STFC, KIT

Important
We are working together
to make things compatible

That’s all

In case I talked too fast

Orpheus

https://orpheus.data.kit.edu

- Gain deep insights
  - into everything

Summary

- No **`ssh`**-daemons (or clients) were hurt in this project:
  - Unmodified SSH Client and Server
  - Backward compatible with: password, ssh-keys, 2nd factor modules, …
- Supported platforms
  - Windows: Putty
  - Mac/Linux: OpenSSH
- Packages: https://repo.data.kit.edu
- Video: https://youtu.be/090D4s0TNaA
- Visit https://ssh-oidc-demo.data.kit.edu to try it yourself

More SSH Approaches

- Multiple different approaches exist
- Smart Shell
  - AWI, SURF
- SSH Certificates
  - DEIC
- PAM Module
  - STFC, KIT

Important
We are working together
to make things compatible

That’s all

In case I talked too fast

Orpheus

https://orpheus.data.kit.edu

- Gain deep insights
  - into everything