Proxy Token Translation Service - internals

Mischa Sallé

msalle@nikhef.nl

EGI Community Forum, Bari

12 November 2015
Focus on interaction

**VO-portal ↔ Master-portal**

(TTS)
Our token: short-lived RFC3820 (VOMS) proxy certificate

Translation from SAML identity to proxy certificate

Access to TTS must be restricted to certain services:

→ delegation scenario: use OpenID Connect
  - TTS acts as OIDC server (Authorization Server and protected Resource)
  - VO Portal acts as OIDC client
  - Use OIDC access token to obtain proxy certificate
Also need End-Entity Certificate (EEC)

Cache EEC in MyProxy credential store behind TTS

CILogon portal-delegation scenario (http://goo.gl/VnMKXS)
  - Uses OpenID Connect for MyProxy protocol
  - OIDC server in front of a MyProxy Online CA
  - TTS acts as OIDC client
  - Uses OIDC access token to obtain End-Entity Certificate

→ Use protocol and OpenID Connect server twice!
End Entity Certificate:
- produced using MyProxy online CA
- OIDC4MP DS is OIDC server
- TTS/Master Portal is OIDC client
- EEC cached in MyProxy credential store

Proxy Certificate (our token):
- produced using EEC in MyProxy credential store
- TTS/Master Portal is OIDC server
- VO Portal is OIDC client
- proxy is retrieved and used by VO portal
TTS Overview: getting a certificate

Architecture

Mischa Sallé (Nikhef)
TTS Overview: cmdline via ssh backdoor

Architecture

Mischa Sallé (Nikhef)
TTS Overview: using OIDC only

Architecture

VOMS

Workmode, UI, Laptop

VO Portal

Delegation Server

OIDC4MP Client

/authorize

OAuth

MyProxy Server

SSH Server

LDAP

/putkey

OIDC-1

SAM/LWAYF

IdP

Delegation Server

Online CA

MyProxy Server

Mischa Sallé (Nikhef)
Bonus features

- **OpenID Connect server:**
  - Reuse OIDC4MP server for pure OpenID Connect
  - SAML-to-OIDC token translation service (not difficult in itself)
  - Broader use for Master Portal

- **SSH backdoor for commandline access:**
  - VO portal: SSH pubkey upload (similar to GitHub, CERN)
  - Master Portal (TTS): store in LDAP
  - cron-job: `authorized_keys` with fixed command (myproxy-logon wrapper)
  - user obtains proxy using SSH-Agent
  - No need for ECP, Moonshot, custom passwords etc.
More Bonus features

- Smooth transition from PUSP:
  - MyProxy CA not much different from MyProxy credential store
  - Can use robot cert+key instead of CA cert+key
  - Few simple changes in config of MyProxy CA

→ produce PUSP instead of EEC

- Based on well-maintained and proven software:
  - Production software, widely used in US
  - Actively developed
  - Maintainers are part of AARC
  - Easy to replace components (modular setup)
Next steps

Implementation Master Portal:

- minor adaptations to profile (already agreed upon): /getproxy endpoint

- extra OIDC server servlet inside Master portal
  - /getproxy endpoint
  - behind /authorize endpoint

- /authorize endpoint
  - first server servlet then client servlet
  - flow for pure OIDC, probably using different scope

- implement SSH key upload: /putkey endpoint?
Work in progress but looking good!

Based on AARC-SA1 pre-pilot work

Combining existing blocks, minimal glue

Many thanks to Tamas Balogh (doing a lot of the hard work)
Some References

- Our setup: https://wiki.nikhef.nl/grid/CILogon_Pre-Pilot_Work
- OpenID Connect for MyProxy: http://goo.gl/VnMKXS
- CILogon docs: http://www.cilogon.org/portal-delegation
- MyProxy: http://grid.ncsa.illinois.edu/myproxy/
  - OA4MP: http://grid.ncsa.illinois.edu/myproxy/oauth/
  - protocol: http://grid.ncsa.illinois.edu/myproxy/protocol/
- VOMS: e.g. http://italiangrid.github.io/voms/
- ssh authorized_keys: man sshd