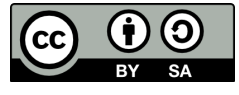
Open (AuthZ) design questions for a scalable and secure federation of storage resources

Francesco Giacomini – INFN
(Software Development @CNAF)

16th SIG-CISS
Federated Storage and Storage Infrastructure Workshop
Bologna – 3-4 December 2025



Our experience

- Authentication/Authorization services
  - INDIGO IAM, *Community AAI* based on OAuth/OpenID Connect technology
  - Virtual Organization Membership Service, *Community AAI* based on X.509 technology, being phased out after 20+ years
  - Argus, Authorization Policy Engine based on XACML, now dismissed
- Storage services
  - StoRM, storage resource manager for disk and tape systems, implementing SRM, WebDAV, WLCG Tape REST API specifications
- Infrastructures
  - World-wide LHC Computing Grid (WLCG)
  - European Grid Infrastructure (EGI)
  - INFN Data Cloud
  - EOSC in the future?

Context

- What is a (storage) federation?
  A federation is an agreement among administratively independent partners. It implies heterogeneous technologies: file systems, tape systems, networks, products, databases, programming languages, performance trade-offs, deployment choices, . . . , exposed through a common, possibly standard, interface.

Context

- What is a (storage) federation?
  A federation is an agreement among administratively independent partners. It implies heterogeneous technologies: file systems, tape systems, networks, products, databases, programming languages, performance trade-offs, deployment choices, ..., exposed through a common, possibly standard, interface.

In this contribution:

- Focus on batch processing, not on Web interaction
- Focus on Authorization, not Authentication
- Use of the OAuth model, with AuthZ based on JWT Access Tokens

Roles vs Capabilities

- Two fundamental authorization models, based on the kind of information included in the authorization token
  - access is granted based on information about subject roles, e.g. identity and group membership
  - access is granted based on information about what actions can be performed

Roles vs Capabilities

- Two fundamental authorization models, based on the kind of information included in the authorization token
  - access is granted based on information about subject roles, e.g. identity and group membership
  - access is granted based on information about what actions can be performed
- Where are AuthZ policies processed?
  - With role-based AuthZ, the main policy decision point is the resource server
  - With capability-based AuthZ, the main policy decision point is the token issuer, with a residual role by the resource server
  - In both cases policies are edited in multiple places (at least by resource owners and community administrators) and there are significant challenges in moving policies around and integrate them in a coherent way where needed
  - For capabilities, additional specification work needs to be done to agree on how to express permissions

Which claims to include in the token?

- Claims determine the authorization outcome at the resource server
- Should the resource server perform the so-called offline validation of the token and find all needed claims therein or should it rely on token introspection, either direct or proxied?
- Offline validation alone requires that all needed claims are in the token, with constraints for example on its size; relying on introspection allows more flexibility
- Introspection adds latency and risk of failure to the request
- How to express the claims? different solutions/profiles (e.g. AARC, WLCG) are available, shaped by different needs/priorities, with attempts to reconcile them

Avoidance/mitigation of token abuse

- How to cope with stolen tokens? How to ban a user or a client?
- Minimize storage of usable tokens, e.g. in database, filesystem, log files (!)
- Constrain tokens as much as possible, in terms of duration, audience, scopes, clients, ...
- Use token exchange instead of reusing the original token in multi-service requests (e.g. user to Rucio to FTS to storage resource)
- Provide mechanisms to revoke tokens and suspend users/clients; then rely on introspection and/or on the distribution of a list of revoked tokens and suspended users/clients

Scalability/availability of the token issuer

- The token issuer is a single point of failure. How to make it able to cope reliably with the expected load of requests?
- High-Availability deployments help
- Geographic distribution of the service, possibly with anycast routing, may be possible, but it's far from trivial.
- Approaches that increase security, such as systematic reliance on token introspection, token exchange and constrained tokens, also increase the number of tokens needed to perform an operation and consequently the load on the token issuer
- Delegating token issuing from the main, trusted issuer to other services, e.g. Rucio for all data management operations, is worth exploring, especially when OpenID Federation becomes mainstream

Integration with computing

- How can a computational job obtain a token to access/download/upload data?
- Note that such an action can happen well beyond the typical lifetime of the original token
- Someone has to make a fresh token available to the job when needed
- Solutions exist, typically integrating the batch system with a vault-like service, that takes care of always keeping a fresh token available to the job
- Are there alternatives? Could pre-signed URLs be a possibility?

Backwards compatibility

- We don't have the luxury to start from scratch: users, infrastructures, services, protocols, ... already exist and process PBs of data daily
- How do you transition to a new world, keeping the existing one running?
- For example, X.509 certificates and proxies with VOMS extensions have been the AuthN/AuthZ foundation for WLCG for 20+ years. How can they be seamlessly replaced by OAuth tokens?
  - It can be done, but it takes a long time

How INDIGO-IAM helps

- INDIGO-IAM is an OAuth Authorization Server and OpenID Connect Provider, designed and implemented to satisfy the needs of the scientific community
- We try to make it flexible enough to be deployed in configurations that could satisfy any reasonable mix of options
  - (proxied) token introspection and offline validation
  - token exchange
  - multiple token profiles
  - roles and capabilities
  - token constraints
  - high availability
  - policy engine
  - VOMS compatibility
- Ongoing work to adopt Open Policy Agent (also in StoRM) and to support OpenID Federation