Enriched namespace to support content-aware authorization policies

Riccardo Zappi (INFN-CNAF)

CHEP'07 - Victoria, BC, Canada

06 September 2007

About this talk

- Problem Statement
- Namespace, Access Control and Metadata
- Attribute-based Authorization
- Our Idea
    - which are the elements?
        - VOMS, File and Metadata Catalog, SKOS, XACML and G-PBox
    - putting all together
- Conclusions

Problem Statement 1/2

There is an explosion in the amount of available data.

- Petabyte scale data sets are coming online
- Petabyte data sets may be spanned in **millions** of files
- There are many replicas of the same file stored at different physical locations
- The data access has to be totally transparent and secure to the user

The management (definition and enforcement) of the access control to this huge, dynamic and dispersed set of data item is a very hard work.

Problem Statement 2/2

Currently, access control is made specifying the authorizations on everyone of millions of files.
To keep under control the difficulties on the access control management, there are some temporary solutions:

- SOL A Grouping users into categories and **roles** and associate the permissions to the group/role rather than to the single user (similar to RBAC)
- SOL B Grouping files and data set into single directory and associate the permissions to the directory path rather than to the complete file name

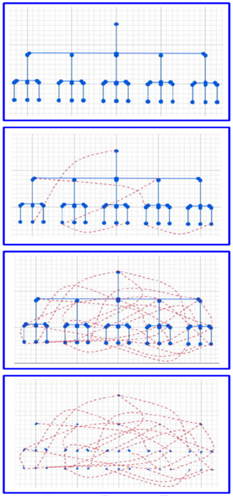
But, are these solutions adequate in large open system like the grid?

Global Namespace

- Hierarchical, named directory trees are nearly identical to the first file system of the 1960s
- Global namespace provides to the user the illusion of a single file system
- The user identifies files with logical file name (LFN)
- Typically, LFNs are based on a naming scheme designed to be easily interpreted, like a directory tree name
- The directory tree is built to reflect a data classification

Limits of Hierarchical namespace

- Strict hierarchical filling schemes fail in many cases
  - A file could live in two or more directories
  - Build symbolic links, aliases and shortcuts
  - Often users forget where their files are
  - Querying file catalog and Searching tools
- Cross-tree linking drawbacks:
  - cross-tree linking are difficult to maintain in the face of file updates and removal.
  - complicates the permissions management and access control enforcement.



Data Access Control

Typically, the access to the files is controlled by **Access Control Lists** (ACL) bound with the global unique identifier (GUID) of file. File owner sets up the permissions according to user name, roles and groups, but

- Which roles exist? Which users are belonging to a specific role?
- VO administrator can modify roles and group composition at any time!
- In any case, permissions must be set for every file (possibly millions of item!) despite the fact that permissions are always the same.

The "SOL A" (use of group/role) does not scale!

Using the Metadata

- In order to moderate the number of permissions to be set, a resource owner could organize its own data items into directories and set up the access control policies only on the directory-path.
- But, traditional hierarchical organization provides a weak and inadequate structure for a meaningful data mining
- **Use of metadata** helps to solve organization of files.
- Metadata allows to describe the contents of data items. They are crucial in order to identify, locate and evaluate quality of the data.

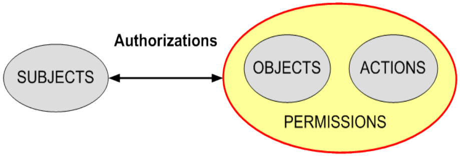
... then permissions bound with directories (”SOL B”) is no more the solution to the problem :(

Semantic layer on Hierarchical Namespace

- Metadata and attribute-based naming allows users to classify each file with multiple attributes.
- Normally, generating and binding accurate attribute values to files is very difficult. Users could tag the same file with different synonyms (Collaborative or Social tagging (folksonomy)).
- In HEP, as well as in every scientific domain, we can use a well specified **thesaurus**, **classification scheme** and **taxonomy**. Hopefully, a same file tagged by different users *should* holds the same attribute values (no more than one interpretation!)

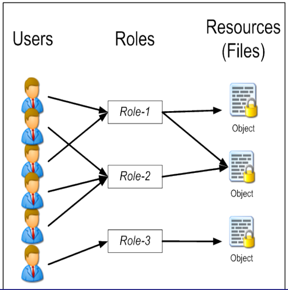
Authorization concepts

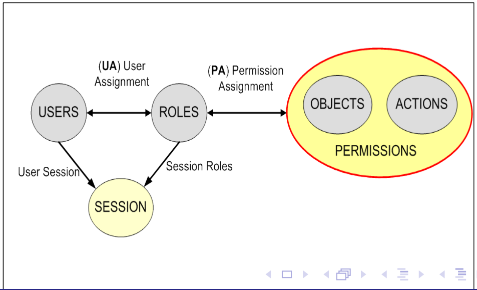
Subject	**active** entities, users or processes trying to access system resources
Object	**passive** entities, resources such as files or system functions
Permission	authorization rules are used to define which subject can access which object and in which way (actions).



Role Based Access Control (RBAC)

- VO users can be organized in a complex, hierarchical structure with groups and **roles**.
- In RBAC model the user access rights are defined in terms of roles.
- Generally, roles are engineered based on the "*principle of least privilege*".

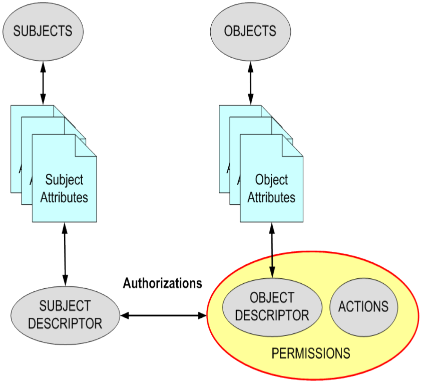




ABAC Model

The basic idea of attribute-based access control (ABAC) is to use subject and object attributes to define permissions.

- **Subject attributes**: groups and roles; personal data (accounting, age, ...)
- **Object attributes**: subjects of the data item; path and file name; ...
- **Permissions** consist of the combination of attributes and conditions like "age>18" and operations on object like "read".
- **Authorizations** are defined between subject descriptor and a permission.



An example of a ABAC Policy

Policy: Users belonging to VO LHCb, group "analysis/review" and with role "Admin" can read every file tagged as *"Analysis results annotated"*

- **Subject attributes** : groups, roles, attributes...
- **Object attributes** : user-defined labels ...
- **Permission** : *perm1* = <Read is allowed for files tagged as "Analysis..">
- **AuthZ policy** : group:/lhcb/analysis/review" | Role="Admin" holds *perm1*.

Our Idea

Apply the ABAC model (with a restriction to the possible values of object and subject attributes) **to the file access control**

- Subject attributes values : groups, role and other attributes VO-defined.
- Object attributes values :
  - System metadata (path, file name, size, etc.)
  - Labels about the file content belongs to a small and well defined set of values (Thesaurus and Taxonomy)
- A language able to declare attribute-based policies
- A framework to manage (writing, distribute, enforce) that policies

Subject attributes and VOMS

VOMS The **Virtual Organization Membership Service (VOMS)** is an *Attribute Authority* that issues attributes for users.

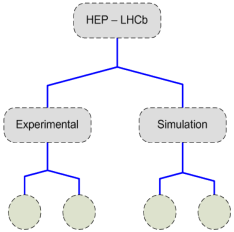
- VOMS uses X.509 identity and proxy certificates
- It encodes the position of the holder inside the Virtual Organization (VO)
- VOMS issues attributes for users (groups, roles and attributes)
- It is being extended to support SAML protocol and assertions
- Currently it is the most widely-used attribute authority service, and is a de-facto standard for AA in production Grids

Thesaurus, Taxonomy and SKOS

SKOS Simple Knowledge Organisation System(s)

- W3C 2nd Public Working Draft, work in progress
- probably Recommendation within 1-2 years

- **SKOS** is an application of RDF (Resource Description Framework, W3C Standard) that allows you to construct a simple hierarchy and/or network of concepts.
- **SKOS** is about declaring and publishing taxonomies, thesauri or classification schemes, for use in a distributed, decentralised information system.



SKOS allows Simple subject- or topic-oriented metadata

Attribute-based Authorization Policies and XACML

XACML (eXtensible Access Control Markup Language) is an OASIS standard that defines both a policy language and an access control decision request/response language.

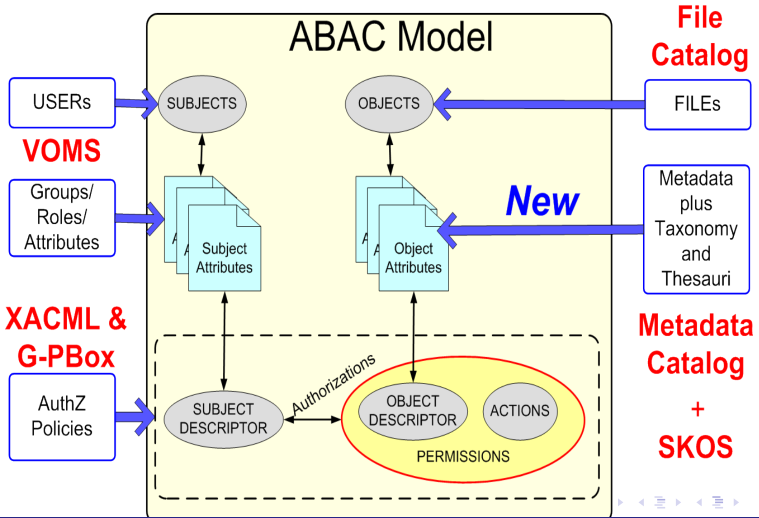
- It is a language based on the XML
- Policies are divided into *targets*, *rules*, *conditions* and *obligations*
- Targets, rules, and conditions act on **attributes** which may be part of the original request, or may be retrieved based on defined *Attribute Authorities* (e.g. VOMS and an hypothetic SKOS AA)

Policies Definition and Distribution with G-PBox

G-PBox is an authorization framework for Grid environments based on XACML.

- G-PBox allows VO and Site admins to write policies for resources/services.
- It is composed by a **server**:
  - which contains the Policy Decision Point (PDP), policy repository, etc..
- and by an administrational **Graphical User Interface** (GUI)
  - It represents a Policy Administration Point (PAP).
  - It allows the definition of complex XACML policies.
  - It communicates to VOMS servers to retrieve subject *attributes* used within policies.
  - It could be extended to communicate to SKOS servers to retrieve taxonomies and thesauri.

Putting all together



Conclusion

- Data sets on the order of petabytes may be spanned in millions of files.
- Manage authorization on huge number of data items is a big problem.
- Classic methods do not works well in open large system like the grid.
- Use of metadata to organize data in a meaningful way is becoming common.
- We propose the introduction of a ulterior metadata to implement an ABAC authorization model.
- Tagging files with labels taken from the domain thesaurus allows a content-aware classification of data items.
- We show a possible high level architecture to implement our idea using technologies already in use.