VOMS, an Authorization System for Virtual Organizations

R. Alfieri[1], R. Cecchini[2], V. Ciaschini[3], L. dell'Agnello[3], Á. Frohner[4], A. Gianoli[5], K. Lőrentey[6], and F. Spataro[3]

[1] INFN and Department of Physics, Parma
[2] INFN, Firenze
[3] INFN, CNAF
[4] CERN, Geneva
[5] INFN, Ferrara
[6] ELTE, Budapest

Abstract. We briefly describe the authorization requirements, focusing on the framework of the DataGrid and DataTAG Projects and illustrate the architecture of a new service we have developed, the Virtual Organization Membership Service (VOMS), to manage authorization information in Virtual Organization scope.

1 Introduction

Authorization plays a key role in the process of gaining access to resources in a computational grid [1].

As for authentication, it is not feasible to administer authorization information on a local site basis, since users have normally direct administrative deals only with their own local site and with the collaborations they work in, but not, generally, with other entities.

It is convenient to introduce the following concepts:

– **Virtual Organization** (**VO**): abstract entity grouping Users, Institutions and Resources (if any) in a same administrative domain [2];
– **Resource Provider** (**RP**): facility offering resources (e.g. CPU, network, storage) to other parties (e.g. VO's), according to specific "Memorandum of Understanding".

From the authorization point of view, a grid is established by enforcing agreements between RP's and VO's, where, in general, resource access is controlled by both parties with different roles, and indeed the main difficulty is to clearly separate these two roles. To solve this apparent dualism, we can classify the authorization information into two categories:

1. general information regarding the relationship of the user with his VO: groups he belongs to, roles he is allowed to cover and capabilities he should present to RP's for special processing needs;

2. information regarding what the user is allowed to do at a RP, owing to his membership of a particular VO.

We think that the first kind of information should be contained in a server managed by the VO itself, while the second is probably best kept at the local sites, near the resources involved and controlled by some kind of (extended) Access Control Lists (ACL).

In this note we briefly describe the authorization requirements, focusing on the framework of the DataGrid and DataTAG Projects [3–5], and illustrate the architecture of a new service we have developed, the Virtual Organization Membership Service (VOMS), to manage authorization information in VO scope.

The VOMS architecture uses the authentication and delegation mechanisms provided by the Globus Toolkit Grid Security Infrastructure (GSI) [6, 7].

1.1 Authorization Requirements

Authorization, as stated before, is based on policies written by VO's and their agreements with RP's, that enforce local authorization. In general a user may be member of any number of VO's and his membership must be considered as a “reserved” information.

On the other hand, a VO can have a complex structure with groups and subgroups in order to clearly divide its users according to their tasks. Moreover, a user can be a member of any number of these groups.

A user, both at VO and group level, may be characterized by any number of roles and capabilities; moreover roles and capabilities may be granted to the user indefinitely or on a scheduled time basis (e.g. a certain user of the CMS collaboration is granted administrative role only when he is ”on shift”) or on a periodic basis (e.g. normal users have access to resources only during working hours).

The enforcement of these VO-managed policy attributes (group memberships, roles, capabilities) at local level descends from the agreements between the VO and the RP's, which, however, can always override the permissions granted by VO (e.g. to ban unwanted users). As a consequence, users must present their credential to RP's (and not just the authorization info).

1.2 VO Structure

Users of a VO are organized in groups which in general form a hierarchical structure with the VO itself as the root; the management of a group can be delegated. Excluding the root, each group may have, in general, several ancestors; note that we cannot have cycles in this structure (i.e. a group which is subgroup of itself). Thus we can represent the VO structure with a Direct Acyclic Graph (DAG)[7].

Users are normally contained in subgroups: for a user being member of a particular group G implies that he is also contained in all ancestor groups, even if it not explicitly stated, up to the root (i.e. in all groups contained in the paths from G to the root).

[7] The groups are the vertices of the graph and the subgroup-group relationships are the oriented edges.

Users are also characterized by roles they can cover in a group or at VO level (but in our model the VO is functionally equivalent to a group) and capabilities (properties to be interpreted by the local sites, e.g. ACL's). Roles are inherited by group members from ancestor groups (i.e. if a user as a role in a group and if he is member of one of its subgroups, he covers the same role in the subgroup), while the opposite is not generally true. The same inheritance rule applies for capabilities.

In conclusion, within this model, if a user U is member of the groups $\{\mathbf{G}_1,\ldots,\mathbf{G}_n\}$, noting with the triplet $(\mathbf{G}_k,\mathbf{R}_k,\mathbf{C}_k)$ the membership, roles and capabilities of U relatively to the group $\mathbf{G}_k$, the complete authorization information about U is formed from the set $(\mathbf{G}_1,\mathbf{R}_1,\mathbf{C}_1),\ldots, (\mathbf{G}_n,\mathbf{R}_n,\mathbf{C}_n)$.

1.3 Authorization status in EDG

The Authentication and Authorization methods adopted by the EDG are based on the Globus Toolkit's Grid Security Infrastructure (GSI) [7].

In EDG, as originally in Globus, to access the Grid, the user first creates a proxy certificate (via grid-proxy-init procedure) that is then sent to the requested resources in order to access them.

In EDG Test-bed 1 each VO maintains information about its users in a LDAP server; each user is member of some groups of the VO. Note that, in the current implementation, subgroups, roles and capabilities are not supported; hence a differentiation among users is only manageable at the local sites. The RP's, periodically (e.g. daily) querying the LDAP servers, generate a list of VO users (in case banning unwanted entries or allowing non-VO users) and map them to local credentials (the so-called "grid-mapfile") granting users the Authorization to access local resources.

In EDG, the front-end of the farm (the *Gatekeeper*) has been modified and access these Authorization data via the Local Credential Authorization Service (LCAS) [10].

The main missing features of this architecture are flexibility and scalability. No roles, subgroups memberships and any other user peculiarity are supported. Moreover, the use of a RP-based database (i.e. the grid-mapfile), periodically updated, hardly scales in a production environment with a large number of users, each, potentially, with his groups, roles and capabilities, whereas in the test-bed the users situation is almost static, and user policy is very simple.

The solution, in our opinion, is to let users present the authorization data as they try to access the local resources (i.e. shifting from pull to push model); on the other hand we suspect that LDAP protocol is not the best choice to sustain the burden of a potentially high number of complex queries. To address these issues, we have developed, on a completely new basis, the VOMS system.

2 The VOMS System

The server is essentially a front-end to an RDBMS, where all the information about users is kept.

The VOMS System is composed by the following parts:

– **User Server**: receives requests from a client and returns information about the user.

– **User Client**: contacts the server presenting a user's certificate and obtains a list of groups, roles and capabilities of the user.
– **Administration Client**: used by the VO administrators (adding users, creating new groups, changing roles, etc...)
– **Administration Server**: accepts the requests from the clients and updates the Database.

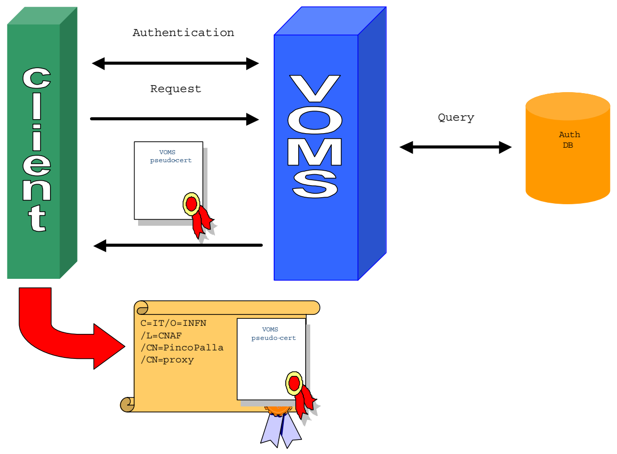

Fig. 1. The VOMS system

2.1 Operations

User part

One strong requirement we faced with, was to disrupt as little as possible – from the user's standpoint – the creation of the user proxy certificate [14]. To achieve this we have added a command (`voms-proxy-init`) to be used in place of `grid-proxy-init`. This new command produces a user's proxy certificate – like `grid-proxy-init` – but with the difference that it contains the user info from the VOMS server(s). This info is returned in a structure containing also the credentials both of the user and of the VOMS server and the time validity. All these data are signed by the VOMS server itself. We call this structure a "Pseudo-Certificate" (in the next release it will become an Attribute Certificate [8, 9]).

The user may contact as many VOMS's as he needs.

In order to use the authorization information, the *Gatekeeper*, in addition to normal certificate checking, has to extract the additional information embedded in the proxy (the Pseudo-Certificate). This can be easily done with an appropriate LCAS plug-in [10]. However, as the VOMS info are included in a non critical extension of the certificate, this can be used even by “VOMS-unaware” *Gatekeepers*, thus maintaining compatibility with previous releases.

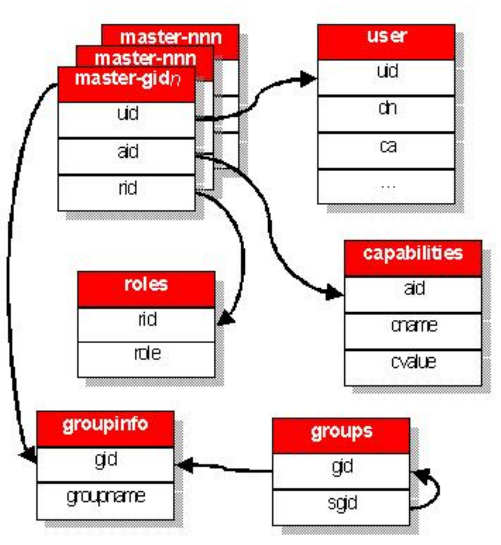

Fig. 2. The VO structure

Administration

The administrative clients (GUI and CLI) share a common server to modify the database. The server can be reached by the SOAP protocol, so that it can be easily converted into an OGSA service. The server consists in three sets of routines, grouped into services: the **Core**, which provides the basic functionality for the clients; the **Admin**, which provides the methods to administrate the VOMS database; the **History**, which provides the logging and accountability functionality.

All tables in the database have a `createdBy` and a `createdSerial` colums. The former contains the id of the requester of the operation that created this record. The latter contains a database-wide unique, ordered serial number – incremented for each modification on the database – that identifies the operation (it is a transaction id).

Copies of deleted and modified rows are kept in a corresponding archive table. Archive tables have the same scheme as data tables, except for two additional columns: `deletedBy`, the administrator who deleted the record, and `deletedSerial`, the transaction number of the operation.

By keeping all expired data in the database, we can conveniently, efficiently and accurately answer such questions as “Was user U in group G at time T?”

2.2 Security Considerations

The VOMS server does not add any security issues at user level since it performs the usual GSI security controls on the user’s certificate before granting rights: it must be signed by a “trusted” CA, be valid and not revoked.

On the other hand, even compromising the VOMS server itself would be not enough to grant illegal access to resources since the authorization data must be inserted in a user proxy certificate (i.e. countersigned by the user himself). Hence the only possible large scale vulnerabilities are denial of service attacks (e.g. to prevent VO users to get their authorization credentials).

The main security issue about proxy certificates is the lack of a revocation mechanism; on the other hand these certificates have short lifetimes (12 hours, typically).

3 Related Works

In this paragraph, we will briefly compare the VOMS system with some analogous systems, namely the ”Privilege and Role Management Infrastructure Standards Validation” (Permis), Akenti and the ”Community Authorization Server” (CAS).

3.1 VOMS vs. Permis

Permis[11], implementing an RBAC (Role Based Access Control) mechanism, has been considered as an alternative to VOMS. We found two major differences.

The first major difference is that in Permis the Attribute Certificates (AC) are kept in a single AC repository, from which certificates are requested from the engine after an user has been successfully authenticated.

On the contrary, VOMS distributes these AC’s to the users themselves, allowing a much greater flexibility. For example, with VOMS a user who is a member of several groups and holds several roles can actually choose how much information about himself he may want to present to a site. It is also possible to obtain and present at the same time information on more VO’s, a useful characteristic in case of collaborations between VO’s.

The second major difference is the policy engine, where Permis is really powerful, because it can take a properly formatted policy file and make decisions based on the content of the file and the AC’s it receives. On the contrary, VOMS does not focus on this problem, and it leave the interpretation of the AC’s to other components (i.e. to local sites, namely to LCAS).

We think that the Permis approach, while perfect in the case of completely separated organizations, may lead to problems in a VO-oriented environment. In fact, since Permis is essentially a policy engine, it is best kept on the local sites where there are the resources it controls. Consequently, having multiple RP’s in grid, it should be hard to maintain consistency among the various repositories; on the other hand, having a central one would not hold clear advantages against the VOMS solution, and, in our opinion, we would loose all the flexibility of the latter.

In conclusion, in our opinion VOMS and Permis are complementary: VOMS as a AC issuer, and Permis (slightly modified in its AC gathering) as an policy engine.

3.2 VOMS vs. CAS

CAS[12] has been developed by the Globus team to solve the same problem tackled by VOMS in EDG.

In our opinion, there are two major differences between CAS and VOMS.

The first is that CAS does not issue AC's, but whole new proxy certificates with the CAS server Distinguish Name as the subject; the authorization information is included in an extension.

As a consequence, when a service receives this certificate, it cannot effectively decide who the owner is without inspecting the extension. This means that existing services, in Globus-based grids, would need to be modified to use a CAS certificate; on the contrary using VOMS, since it adds the AC's in a non-critical extension of a standard proxy certificate, does not require this kind of modification to the services.

The second major difference is in the fact that CAS does not record groups or roles, but only permissions. This means that the ultimate decision about what happens in a farm is removed from the farm administrator and put in the hands of the CAS administrator, thus breaking one of the fundamental rules of the grid: the farm administrator has total control about what happens on his machines.

3.3 VOMS vs. Akenti

Akenti[13] is an AC-based authorization system.

In our opinion, there are three major differences between Akenti and VOMS.

The first is that Akenti does not use true AC's since their definition and description do not conform the standard [8].

The second is that Akenti is targeted on authorizing accesses on web resources, and particularly web-sites. This means that it is completely unfeasible to use it for other needs, for example in a VO.

The third is that Akenti does not link identities with groups or roles, but with permissions. This is done on the resource side, not removing the control from the resource itself, like CAS does; on the other hand, not having an intermediary like VOMS (or even CAS) will surely lead to fragmentation and inconsistencies between the permissions.

4 Future Developments

Future developments will include use of Attribute Certificates, replica mechanisms for the RDBMS, more sophisticate time validity for the VOMS certificates and subgroups.

[8] At present nor VOMS uses standard AC's, but this will be changed in the next (production) release

References

1. Foster, I. and C. Kesselman (eds.), The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (1999)
2. I. Foster, C. Kesselman and S. Tuecke, The Anatomy of the Grid, International Journal of High performance Computing Applications, **15**, 3 (2001)
3. The DataGrid Project: http://www.edg.org/
4. The DataTAG Project: http://www.datatag.org
5. iVDGL - International Virtual Data Grid Laboratory: http://www.ivdgl.org/
6. The Globus Project: http://www.globus.org/
7. Grid Security Infrastructure: http://www.globus.org/security/
8. R. Housley, T. Polk, W. Ford and D. Solo, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC3280 (2002)
9. S. Farrel and R. Housley, An Internet Attribute Certificate Profile for Authorization, RFC3281 (2002)
10. Architectural design and evaluation criteria: WP4 Fabric Management, DataGrid-04-D4.2-0119-2-1 (2001)
11. Privilege and Role Management Infrastructure Standards Validation: http://www.permis.org/
12. L. Pearlman, V. Welch, I. Foster, K. Kesselman and S. Tuecke, A Community Authorization Service for Group Collaboration, IEEE Workshop on Policies for Distributed Systems and Networks (2002)
13. http://www-itg.lbl.gov/Akenti/
14. S. Tuecke, D. Engert, I. Foster, V. Welch, M. Thompson, L. Pearlman and C. Kesselman, Internet X.509 Public Key Infrastructure Proxy Certificate Profile, draft-ggf-gsi-proxy-04 (2002)