ISTITUTO NAZIONALE DI FISICA NUCLEARE

CNAF–Bologna

INFN/TC-04/18
22 December 2004

AN ABSTRACT MODEL OF THE VIRTUAL ORGANIZATION MEMBERSHIP SERVICE

Sergio Andreozzi, Vincenzo Ciaschini, Luca dell'Agnello

INFN-CNAF, Viale Berti Pichat 6/2, 40127 Bologna, Italy
email: andreozzi,ciaschini,luca@cnaf.infn.it

Abstract

Grid systems involve the coordination of resources and services that are not subject to centralized control, span multiple trust domains and are accessible by a large number of users. They require a functionality for mapping users and credentials that exist at the virtual level to users and credentials that exist at the physical level. In this paper, we present a rigorous modeling of the Virtual Organization Membership Service (VOMS), a credential-based access control system. It enables the management and secure distribution of roles, groups and capabilities of users at the virtual level by improving both scalability and flexibility aspects.

*Published by **SIS–Pubblicazioni***
Laboratori Nazionali di Frascati

1 Introduction

Grid systems follow a new paradigm of distributed computing that enables the coordination of resources and services that are not subject to centralized control. These resources are heterogeneous, geographically dispersed, and can dynamically join and leave virtual pools. A Grid system may involve multiple trust domains and resources are assigned to users by means of an explicit assignment functionality [27]. These characteristics imply new requirements for the access control model, i.e, authentication and authorization mechanisms [26].

The current production-quality Grid systems are based on the Globus Toolkit version 2 [17]. The several components have been developed from several sources, with published and stable interfaces. In this scenario, with a larger number of users (hundreds) and sites (tens) a strong requirement is that authorization at each resource must be managed by some automated procedure, which derives local policy from one or more central, manually-managed sources of authorization on a 'Virtual Organization' (VO) basis (collection of individuals and institutions that is defined according to a set of resource sharing rules [19]). VOs generally share resources and establish agreements with general facilities, called Resource Providers (RP) offering resources (e.g., CPU, network, and storage).

In this paper, firstly we summarize the prior art on access control models with particular concern of the authorization schemes existing before the Grid paradigm (Section 2). Then, we present the identified requirements as regards the access control as needed by Grid systems (Section 3). Subsequently, we present the first attempt of a rigorous model of the Virtual Organization Membership Service (VOMS) [6], a tool for the management of Grid user credentials on a VO base (Section 4). Finally, we comment by placing the VOMS in the larger picture of access control models and we describe relevant parallel efforts in the area (Section 5).

2 Background on Access Control Models

In this section, relevant works in the area of access control are presented. We start by presenting the main authorization schemes (Section 2.1) and then we review proposals in the area of distributed systems, with particular attention to credential-based models (Section 2.2).

2.1 Authorization Models

The main authorization schemes are presented, starting from the historical Mandatory Access Control (MAC) model born in the military sector, moving to the Discretionary

Access Control (DAC) model born in the commercial sector and finally discussing the Role-Based Access Control (RBAC) model.

2.1.1 Mandatory Access Control

Early work in the area of access control modeling was stimulated by the military sector in the 70s. The primary concern was to permit simultaneous access to information with different sensitivities by users with different security clearances. The United States Department of Defense defines Mandatory Access Control (MAC) as "a means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (e.g., clearance) of subjects to access information of such sensitivity" [5]. In general, in MAC models the system controls access to the objects according to a security policy, and the enforcement of the policy is independent of users'discretion or actions.

An important family of policy models to be used in a MAC system is the Multi Level Security (MLS) policy, which formalization is due to Bell and LaPadula [8] (republished in [9]). In their work, often referred as Bell-LaPadula or BLP, the following elements are defined: subjects, objects, access rights and security levels. Subjects are active entities that want to access stored information or objects (sets of passive and protected entities). The access rights contain mainly two operations: 'read' and 'write', while a security level is a tuple consisting of a classification and a set of categories. The set of classifications contains names ordered by a '$>$' relation (e.g., top-secret, secret, confidential, restricted unclassified and unmarked). The set of categories contains names describing compartments. Each object is associated with a security level denoting its degree of sensitivity. Each subject is associated with a maximum security level and a current security level, which can be changed dynamically if necessary. Two properties are then defined to express the security policy: (1) the 'simple security property', also known as 'no read up', states that no subject may read objects at a higher level than his/her current level; (2) the '*-property', also known as 'no write down', states that no subjects may write to objects at a lower level than their current level. In the BLP, the definition of these security properties causes the system to behave as a MAC.

2.1.2 Discretionary Access Control

The basic pillar behind the Discretionary Access Control (DAC) is that the owner of an object should be trusted to manage its security. The United States Department of Defense defines Discretionary Access Control (DAC) as a "a means of restricting access to objects based on the identity of subject and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) to any other subject” [5].

The first abstract model has been defined by Lampson [23] in the area of operating systems. He introduced the concept of ‘Access Control Matrix’, a two-dimensional matrix with a row for each subject and a column for each object. An element in the matrix specifies the access rights that a subject has on an object. When either subjects or objects are numerous, the matrix becomes very large and it is sparse. In order to have efficient implementations, different strategies can be applied. The first one is to store the matrix by rows as ‘capability lists’. A capability is a tuple of (object, access rights) and it is stored for each subjects. The second strategy is to store the matrix by columns as ‘access control list’ (ACL). An ACL entry is a tuple of (subject, access rights) and it is stored for each object. Another possible strategy is the one in use in UNIX file system. Subjects have been reduced to three: object owner, group and everyone in the system. The access rights is a 3-bit value encoding read, write and execute operations.

2.1.3 Role-Based Access Control

The previous access control models for the military (MAC) and commercial (DAC) sectors do not naturally reflect higher-level organization policies. For a wide range of use cases, protected objects are not generally owned by users, but rather by the organizations or agencies to which users belong. Moreover, access requests are typically made by a user as acting a certain role, thus access control decisions are influenced by the role to which a set of duties and responsibilities are bound. Based on these considerations, a more appropriate model has been defined by Ferraiolo and Kuhn [16]: the Role-Based Access Control (RBAC). Since their seminal paper written in 1992, several evolutions of the model have been proposed. In this section, we mainly refer to the RBAC reference model, recently published as an American National Standard by the American National Standards Institute, International Committee for Information Technology Standards (ANSI/INCITS) [7].

The basic elements are: ‘objects’, they can be any system resource subject to access control, such as a file, printer or database record; ‘operation’, an executable image of a program, which upon invocation executes some function for the user; ‘permission’, an approval to perform an operation on one or more RBAC protected objects; ‘role’, a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role; ‘user’, a human being, although it can be extended to include machines, networks, computer processes or intelligent autonomous agents.

The RBAC reference model is defined in terms of four model components. The

‘Core RBAC’ (1) defines a minimum collection of RBAC elements, element sets and relations in order to completely achieve a Role-Based Access Control system. This components includes user-role assignment and permission-role assignment relations. In addition, it introduces the concept of role activation as part of a user’s session within a computer system. The ‘Hierarchical RBAC’ (2) adds relations for supporting role hierarchies, that is partial orders defining a seniority relation between roles, whereby senior roles acquire the permissions of their juniors and junior roles acquire users of their seniors). In addition, Hierarchical RBAC goes beyond simple user and permission role assignment by introducing the concept of a roles set of authorized users and authorized permissions. The Static Separation of Duty Relations (3) adds exclusivity relations among roles with respect to user assignments. Because of the potential for inconsistencies with respect to static separation of duty relations and inheritance relations of a role hierarchy, this component defines relations in both the presence and absence of role hierarchies. The ‘Dynamic Separation of Duty Relations’ (4) defines exclusivity relations with respect to roles that are activated as part of a user’s session.

The ANSI RBAC defines also an administrative functional specification for performing administrative queries and system functions for creating and managing RBAC attributes on user sessions and making access control decisions. This standard also provides a unified reference model for the features that have achieved acceptance in the commercial marketplace. Although the level of maturity is good, there are still relevant application requirements not addressed. One of the missing features is related to the temporal dimension that roles may have. This issue has been recently addressed with the Generalized Temporal RBAC (GTRBAC) model [22]. It incorporates a set of language constructs for the specification of various temporal constraints on roles (both periodic and duration), including constraints on their activations as well as on their enabling times, user-to-role assignment and permission-to-role assignments. In particular, GTRBAC makes a clear distinction between role enabling and role activation. An enabled role indicates that a user can activate it whereas an activated role indicates that at least one subject has activated a role in a session. The notion of separate activation conditions is particularly helpful in large enterprise with several hundred users belonging to the same role. It helps in the selectively management of role activations at the individual user level.

2.2 Distributed Access Control

Several modern distributed systems rely on the general idea of using credentials in provision of distributed access control. A credential is a digital and digitally signed document that asserts a binding between a principal, represented by a public key, and some property.

Example of the meaning of a stated property are an identity, non-identifying aspect of a user (e.g., a skill or a profession), a granted capability for a service. Ideally, the signing issuer of a credential is responsible for the correctness of the assertion of certificates. Any agent inspecting a shown credential, firstly verifies the signature of the credential and secondly evaluates his trust in the issuer. Evidently, credentials can be exploited for access control in open, distributed and interoperable systems. Local autonomous access decisions for the sake of integrity and confidentiality are based on appropriate interpretations of properties that are extracted from submitted credentials. Depending on how credentials are used, distributed access controls can be grouped into two categories: identity-oriented and key-oriented.

2.2.1 Identity-Oriented Approach

With an identity-oriented approach, the public key is bound to a name (authentication), and then the name is bound to the set of authorizations (authorization). It requires a global naming scheme in order to create unique names for users. Moreover, it requires a trust mechanism for the distribution of public key (e.g., central directories, web of trust). Concerning the authentication, the most widely used infrastructures for the binding from a public key to a name are the X.509 Public Key Infrastructure (PKIX) [21,20] and the Pretty Good Privacy (PGP) [33]. Concerning the authorization, the most prominent standard for binding public keys with authorization is the X.509 Privilege Management Infrastructure (PMI), part of the edition four of X.509 standard (2001). PMI let authorize user requests after a PKI-based authentication. The primary data structure is an X.509 attribute certificate (AC) [15] issued by an attribute authority (AA) to users. The attribute certificate strongly binds a set of attributes describing access privileges to its holder. The AC is signed by the AA, thus providing data integrity and issuer authentication. The root of a PMI is the Source Of Authority (SOA) and can delegates AA. PMI supports DAC, MAC and RBAC models.

The identity-oriented approach is widely deployed in current systems, but it shows some shortcomings. Firstly, it is based on the assumption that a name uniquely identifies a principal. Both uniqueness and general naming strategy can be a problem in large distributed systems, raising problem of scalability and flexibility. Secondly, it assumes the use of previously acquired knowledge about persons in order to make decisions. With the explosion of the Internet, it is likely that one will encounter keyholders who are complete strangers in the physical world and will remain so. In such cases, the keyholder's name is rarely of security interest. A user of a certificate needs to know the existence of granted authorization. Finally, X.509 PKI is based on a hierarchical trust model, hence driving to single point of failure, single point of vulnerability, limited scalability and poor flexibility.

2.2.2 Key-Oriented Approach

With a key-oriented approach, the use of names is avoided by binding the public key directly to the authorizations. The main assumption is that names are not sufficient for authorization decisions. Moreover, global naming schemes are avoided since are not suitable for large distributed systems. The general approach is that every principal generates its own key pair and is responsible for safeguarding their private key.

There are two relevant works in this category. The first is a merge of the Simple Public Key Infrastructure (SPKI) and the Simple Distributed Security Infrastructure (SDSI), called SPKI/SDSI [14]. In this infrastructure, every principal, including a person, a process or a service, may freely generate a cryptographic key pair and it is identified by its public key. Moreover, every principal can sign and issue certificates using its own private key and a signed certificate can be verified by any principals with the public key of the signer. Certificates can be of three types: authorization certificate (can form chains for transferring from a principal to another), name certificate (binds public key to names) and access control list (it represents the security policy of a service, to be privately stored by the service). Concerning the naming scheme, a concept of local names is supported to give a binding from a key to a human-recognizable name. Local names are defined within the local namespace of a principal, similar to names in a personal address book. Local names are not required to be globally unique, but need to be locally unique for the principal who defines them. A globally unique version of a name could be obtained by linking a local name with its namespace, resulting a linked name.

The second relevant work is KeyNote [10], a trust management system. The concept of trust management systems has been introduced in the PolicyMaker system [11] and is a unified approach to specifying and interpreting security policies, credentials and relationships. Trust-management credentials describe a specific delegation of trust and subsume the role of public key certificates; unlike traditional certificates which bind keys to names, credentials can bind keys directly to the authorization to perform specific tasks. A trust-management system has five basic components: (1) a language for describing ‘actions’, which are operations with security consequences that are to be controlled by the system; (2) a mechanism for identifying ‘principals’, which are entities that can be authorized to perform actions; (3) language for specifying application ‘policies’, which govern the actions that principals are authorized to perform; (4) a language for specifying ‘credentials’, which allow principals to delegate authorization to other principals; (5) a ‘compliance checker’, which provides a service to applications for determining how an action requested by principals should be handled, given a policy and a set of credentials. The trust-management approach has a number of advantages over other mechanisms for specifying and controlling authorization, especially when security policy is distributed over a network or is otherwise decentralized.

3 Access Control Requirements in the Grid environment

The first important consideration to be asserted in a Grid environment as regards access control is that it is strongly based on the concept of Virtual Organization: VOs administer users granting them permissions, establish agreements with resource providers that, in turn, enforce local authorization. Agreement on a common rigorous definition of VO is still missing. In [19], it is described as the set of individuals and/or institutions defined by a set of rules by which resources are shared. In the wider literature, a more rigorous definition is given: "a Virtual Organization is a new organizational form which manifests itself as a temporary or permanent collection of geographically dispersed individuals, groups, organizational units, either belonging or not belonging to the same organization, or entire organizations that depend on electronic links in order to complete the production process" [29].

In the framework of the European DataGrid [1] and DataTAG [2] projects a number of requirements for VO-based access control have been identified: (1) a VO can have a complex and hierarchical structure with groups in order to capture the natural composition of institutions; (2) the VO itself is considered a group and is the ancestor of all other groups; (3) from an administrative point of view, the management of each group can be independently delegated to different administrators; (4) a group administrator can create subgroups and grant administration rights to these subgroups; (5) membership in a group implies membership in at least one of the parents groups; (6) users credential always carry the full list of groups to which a user belong to; (7) VO must be able to define roles; (8) each role carry out a set of privileges; (9) the scope of a role is the group in which it is defined; (10) users should be able to dynamically select the role that they want to receive when using the Grid; (11) users must be associated with capabilities, used to describe special characteristics of users not captured by roles or groups; (12) group membership, role and capabilities can have a temporary validity; (13) the enforcement of the VO-managed policy attributes (group memberships, roles, capabilities) at the resources must reflect the agreements between the VO and the resource owners, however it should be possible for owners to override the permissions granted by VOs (e.g., to ban unwanted users); (14) traceability of user behavior should be possible, therefore users must present their credential to resource providers along with their authorization information.

To the above requirements, we consider the following ones: (1) being able to flexibly trust organizations, meaning that participants in those organizations need not take any special action; (2) the need for dynamic delegation, that is allowing an entity A to grant to another entity B the right for B to be authorized with others as if it were A; (3) dealing with dynamic entities, that is the delegated entity can be not only persistent, but also volatile, like a service; (4) dealing with repeated authentication, that is entities may need to be authenticated several times in a short period of time and there should be mechanisms that limit the distribution of pass phrases or similar (e.g., support for single sign-on) [31].

3.1 The Grid Security Infrastructure

In the last years, the Globus Toolkit [4] has emerged as the dominant set of software components for building Grid middleware. An important contribution is the Grid Security Infrastructure (GSI) [18,12,32], providing mutual authentication, message protection, single sign-on and delegation capabilities. It is based on the PKI X.509 [21,20] and the SSL/TSL (Security Socket Layer/Transport Layer Security) protocol [13]. Moreover, it relies on the Generic Security Service Application Program Interface (GSS-API) for the current implementation [24].

In order to satisfy the requirements identified in the previous section as regards the authentication, the concept of Proxy Certificate has been introduced as an enhancement to the standard X.509 PKI [31]. It is a solution that allows users to dynamically create identities for new entities in a light-weight manner, to delegate privileges to those entities and to perform single sign-on, all of requiring only small modifications to the PKIX infrastructure. This approach has been better formalized and consolidated as an IETF RFC, where the term Proxy Certificate has been defined as certificate that is derived from, and signed by, a normal X.509 Public Key End Entity Certificate or by another Proxy Certificate for the purpose of providing restricted proxying and delegation within a PKI based authentication system [30].

4 The Virtual Organization Membership Service Model

In this section, we present the abstract model of the Virtual Organization Membership Service (VOMS), a solution developed in the European DataGrid/DataTAG projects [1,2] to the problem of VO-based distributed access control in a Grid system.

4.1 Reference Model

The first concept to be formalized is the structure of a VO as concerns the group dimension. Given the requirements expressed in Section 3, we choose to represent the VO structure as a Direct Acyclic Graph (DAG) where the groups are the vertexes of the graph and the subgroup-group relationships are the oriented edges. As a special case, we have the VO as group containing all other groups. We start by giving the definition of directed graph.

Definition 4.1 (Directed Graph) *A directed graph G is a pair (V, E), where V is the set of vertexes (or nodes), $E \subseteq (VxV)$ is the set of edges and $e = (v_1, v_2) \in E \iff v_1 \rightarrow v_2$.*

Then, we introduce the property of acyclicity, hence defininining the directed acyclic graph (DAG).

Definition 4.2 (Directed Acyclic Graph/DAG) *A directed acyclic graph is a directed graph with no cycles among the edges.*

Another important property to be added to the VO structure refers to the fact that there should be only one root, that is the VO.

Definition 4.3 (Connected Directed Acyclic Graph/CDAG) *A connected directed acyclic graph is a DAG that has one vertex from which all other nodes can be reached. Such a node is termed the root of the CDAG.*

The last important property to be added is that each node of a CDAG (that is a group of the VO) contains the union of all the users part of the descendant groups.

Definition 4.4 (Group Hierarchy) *Let U be the set of users and $G \subseteq \mathcal{P}(\mathcal{U})$ the set of groups. The Group Hierarchy $\mathcal{GH} = (\mathcal{G}, \mathcal{E} \subseteq \mathcal{G}xG)$ is a CDAG such that $\forall g \in G \Rightarrow g \subseteq \bigcup_{i \in I} g_i$ where $I = \{i \,|\, g_i \rightarrow g\}$ is the set of indexes identifying the edges called 'father of g'.*

After to have defined the group structure, let us introduce the concepts of Role and Capability. We define $\mathcal{R}$ as the set of roles and $\mathcal{C}$ as the set of capabilities. Roles and capabilities have a flat structure. The next concept needed in this model refers to the temporal validity. Let $\mathcal{V}$ be a set of time periods defined as $\mathcal{V} = \{(s, e) | s, e \in \mathrm{TIMESTAMPS}\}$ and Φ a set of periodicals (e.g., *every first of the month, every 36 hours, every day*). Now, we can define $\mathcal{T}$ to be the set of time constraints as follows:

$$\mathcal{T} = \{(v, \phi) | v \in \mathcal{V} \wedge \phi \in \oplus\}$$

Let us also define a function:

$$\text{eval} : (\text{TIMESTAMP}, \mathcal{T}) \rightarrow \{\text{true}, \text{false}\}$$

$$\text{eval}(t, s) = \begin{cases} \text{true} & t \text{ is inside the bounds of } s \\ \text{false} & \text{otherwise} \end{cases}$$

Finally, we define $\mathcal{M}$, the set of user properties, this way:

$$\mathcal{M} = \{(u, g, r, c, t) | u \in \mathcal{U} \wedge g \in \mathcal{G} \wedge r \in \mathcal{P}(\mathcal{R}) \wedge c \in \mathcal{P}(\text{C}) \wedge t \in \mathcal{T}\}$$

This means that a user u, as a member of the group g holds the roles r, has the capability c and all this if and only if the timestamp satisfies t.

4.2 Administration Model

The VO structure as regards the group dimension defined in the previous paragraph is expected to change with time. For instance, users may be added or removed, groups may be created or disbanded, roles may be defined or deleted. In order to address these problems, VO managers are expected to be aware of the organization of the data and to be authorized to modify it. VOMS considers them to be a special kind of users, named administrators, which are allowed to access and modify based on a set of permissions. These permissions include creating users, groups, role, capabilities, time periods, assigning or removing users to groups, roles and capabilities, creating new admins and assigning them permissions. Let $\mathcal{A}$ be the set of administrator and $\mathcal{P}$ be the set of permissions. Then, we have a set $\mathcal{ACL} = \{(a, p) | a \in \mathcal{A} \wedge p \in \mathcal{P}\}$ which defines the permissions given to each administrator.

In the remaining part of this section, we provide more details on the set of operations that are expected to be useful for an administrator and that are currently supported by the existing VOMS implementation. For this purpose, we define $Rights(a, g)$ to be the set of permissions granted to administrator a in the group g. Also, $Grant(a, g, r)$ is a couple $(grant, granter)$ that indicates that the right r on group g to admin a has been granted by admin granter, with or without grant option. Finally, vo is the group that is the ancestor of all other groups and $caller$ is the administrator that invoked an operation.

AddAdmin(a) This operation adds a new administrator a to the structure.

Prereq:	$caller \in \mathcal{A}$
Result:	*a* is now an administrator
Postreq:	$\forall g$, ***Rights(a,g)*** = $\{\}$
	$Rights(caller, vo) \cup (DeleteAdmin(a))$

DelAdmin(a) This operation deletes the administrator a.

Prereq:	$caller, a \in \mathcal{A}$
	$(DeleteAdmin(a)) \in Rights(caller, vo) \vee$
	$\Big((DeleteAdmin(a_1)) \in Rights(caller, vo) \wedge$
	$(DeleteAdmin(a_2)) \in Rights(a_1, vo) \wedge \ldots \wedge$
	$(DeleteRights(a)) in Rights(a_n, vo)\Big)\ a, a_1, \ldots, a_n \in \mathcal{A}$
Result:	a is no longer an administrator
Postreq:	$\forall Grant(c, g, r) = (grant, a)$, the couple is changed to $(grant, \{\})$

Note: As can be seen, this specification does not impact on the right of administrators created, or that received grants, from the removed user. This because in a large enough organization administrators are supposed to change with a certain frequency, and this operation must not interfere with the work of others.

AddGroup(fathers,child) This operation adds group *child* as subgroup to *fathers*, giving to *caller* full rights on *child*.

Prereq:	$(CreateGroup, grant) \subseteq Rights(caller, g)$ for some value of $grant$
Result:	A new group, *child*, has been created as a subgroup of *fathers*
Postreq:	$\forall a \neq caller, Rights(a, child) = Rights(a, father)$
	$\forall r, Grant(caller, child, Rights(caller, child)) = (true, caller)$

AddRole(r) This operation creates a new role r.

Prereq:	$(createrole, grant) \subseteq Rights(caller, vo)$ for some $grant$
Result:	Role r has been created
Postreq:	$\forall r, Rights(caller, r) = r$
	$\forall a \in \mathcal{A}, g \in groups, Rights(a, g)\ also\ has\ GiveRole(r)$

AddUser(u) This operation creates a new user u.

Prereq:	$(createuser, grant) \subseteq Rights(caller, vo)$ for some $grant$
Result:	User u has been created
Postreq:	$Rights(caller, u) = \{r \mid \forall r\}$
	$\forall\, a \in \mathcal{A}, Rights(a, u) = \{(addto\{group, role, cap\})\}$

AddCapability(c) This operation creates a new capability c.

Prereq:	$(createcap, grant) \subseteq Rights(caller, vo)$ for some $grant$
Result:	Capability c has been created
Postreq:	$Rights(caller, c) = \{c \mid \forall c\}$

Grant(a,r,g,grant) Grants the right r to the admin a on the object g and its subgroups, with granting option if $grant$ is true.

Prereq:	$caller$ and a must be administrators themselves
	$r \subseteq Rights(caller, g)$
	$Grant(caller, g, r) = (true, adm)$ for some admin adm
Result:	r is added to $Rights(a, g)$
	$(grant, caller)$ is added to $Grant(a, g, r)$
Postreq:	

Revoke(a,r,g) Revokes the rights r to the admin a on the object g and its subgroups.

Prereq:	a must be an administrator and
	(grant, caller) must be in $Grant(a, g, r)$ or $caller = a$
Result:	The right are revoked recursively
Postreq:	r is removed from $Rights(a, g)$

DeleteGroup(g) This operation deletes group g and all its subgroups.

Prereq:	$DeleteGroup(g) \in Rights(caller, g)$
Result:	Group g is deleted along with its subgroups
Postreq:	$\forall h$ such as that $h \in$ Children(g) and #(Fathers(h)) = 1,
	DeleteGroup(h) has already been called

DeleteRole(r) This operation deletes role r.

Prereq:	$(createrole, grant) \subseteq Rights(caller, vo)$
Result:	Role *r* is deleted
Postreq:	$\forall g, u, p, t$ (u, g, r, p, t) is removed from M, and (u, g, NUL, p, t) is added in its place

DeletePermission(c) This operation deletes capability c.

Prereq:	$deletecap(c) \in Rights(caller, vo)$
Result:	Capability *c* is deleted
Postreq:	$\forall g, u, r, t$ (u, g, r, p, t) is removed from M, and (u, g, r, NUL, t) is added in its place

DeleteMembership(u,g) This operation removes the user u from the group g.

Prereq:	$removeuser(u) \in Rights(caller, g)$
Result:	The user u is removed from group g
Postreq:	(u, g, r, p, t) is removed from M
	Also, Given $H = \{h \| h \prec g \wedge h \prec h1, h1 \in H, \#Fathers(h) = 1\}$,
	(u, h, r, p, t) is removed from M

GiveMembership(u,g) This operation gives to the user u membership in the group g.

Prereq:	$(adduser, grant) \subseteq Rights(caller, g)$
Result:	The user u is added to the group g
Postreq:	$(u, g, NULL, NULL, NULL)$ is added to M

GiveCapability(u,c,g) This operation assigns capability c to the user u in the group g with the role r .

Prereq:	$addcap(c) \in Rights(caller, vo)$
Result:	The capability c is assigned to the user u VO-wide
Postreq:	$(u, g, r, p, t) \in M$ implies that also (u, g, r, c, t) is now added to M

GiveRole(u,g,r) This operation assign to the user u, as a member of group g, the role r.

Prereq:	$addrole(r) \in Rights(caller, g), u \in g$
Result:	The user u now holds the role r in the group g
Postreq:	(u, g, r, NUL, NUL) is now added to M

DeleteCapability(u,c) This operation remove capability c from the user u.

Prereq:	$remcap(u) \in Rights(caller, vo)$
Result:	The user u no longer holds the capability c
Postreq:	$\forall g, r, t\, (u, g, r, c, t)$ is removed from M

DeleteRole(u,g,r) This operation remove roles r from the user u that is a member of the group g.

Prereq:	$remrole(u, g, r) \in Rights(caller, g)$
Result:	The user u no longer holds the capability r
Postreq:	$\forall c, t\, (u, g, r, c, t)$ is removed from M

To the above structure and operations, it is necessary to add the administrator $root$, which always has permission to do anything and cannot be removed. This is needed to prevent the whole structure to become useless due to the possibility for administrators to accidently remove their own rights.

5 Commentary

First generation of Grid systems relying on GSI-based authentication adopted a DAC approach to resource access control by maintaining at each resource a list of authorized users (ACL). The VOMS represents a sensitive improvement in both scalability and flexibility aspects. As regards scalability, it removes the need for storing the complete list of authorized users at each resource. VOs sign contracts with resource providers to let their users access the authorized resources. The VOMS approach removes the need for storing/updating long access list by inserting VO-based credentials in the user proxy certificate. As regards flexibility, role property, group membership and capabilities list are attributes that increase the flexibility of VO managers when carrying out their activities. Finally, the fact that real privileges on physical resources are still managed by resource owners is a key principle that let the VOMS model be adopted by several Grid projects.

Concerning the categorization given in Section 2.1, VOMS can be considered a credential-based system with an identity-oriented approach to access control, since it relies on an enhanced PKIX infrastructure for the distribution of membership data, hence on a global naming scheme.

Parallel efforts to VOMS that are meaningful to be recalled are the Community Authorization Service (CAS) [28] and the PRIMA (Privilege Management, Authorization and Enforcement in Grid Environments) [25]. The first proposal is part of the Globus Alliance project [3] and was born to enables sites for the delegation of the management of a subset of their policy space to the VO. CAS provides a fine-grained mechanism for a VO to manage delegated policy spaces, allowing it to express and enforce expressive, consistent policies across resources spanning multiple independent policy domains. In this sense, it was a central point of management for a VO as regards the accessible resources. In the recent evolution, it adopted the VOMS philosophy of keeping at the resource providers the ultimate authority over their resources.

The PRIMA model is designed for fully distributed operation by supporting the creation of small, transient and ad-hoc communities without imposing the requirement to deploy group infrastructure components like community servers. It consists of a set of three interrelated elements: the first element is a privilege-based security model which defines the nature of how privileges are defined, shared and managed among and between users and system administrators within the bounds of a controlling privilege management policy; the second element is an enforcement model, an implementation-neutral description of how the intent of the security model is achieved; the third element is the enforcement mechanism, it provides a system-specific implementation of the enforcement model; these mechanisms must guarantee that the effects defined by the enforcement model are properly realized on a given system. The main differences between VOMS and PRIMA are: (1) VOMS provides a community centric attribute server that issues authorization attributes to members of the community, while PRIMA is user centric; (2) VOMS uses the PKIX infrastructure empowered by the Proxy Certificate [30] for the privileges distribution, while the PRIMA uses the Attribute Certificate [15]; (3) VOMS is a model for VO-based credential managements and does not model how credentials can be mapped on permissions over resources, while PRIMA provides also components for privileges enforcements at the resource side. In this sense, they are complementary and they both could benefit from a reciprocal collaboration.

6 Conclusion

The Grid paradigm presents new challenges as regards the access control management. After to have presented a summary on the evolution of access control systems, the requirements of a Grid have been identified and presented. The Grid Security Infrastructure (GSI) has been presented as an important building block for the authentication step. As regards the authorization step, this paper presented a rigorous model of the VOMS, a VO-centric credential-based access control system that enables the management and secure distribution of roles, groups and capabilities of users. The VOMS has been also related to parallel efforts in place within the Grid community and differences and similarities have been depicted.

References

[1] *European DataGrid Project*, http://www.edg.org.

[2] *European DataTAG Project*, http://www.datatag.org.

[3] *The Globus Alliance*, http://www.globus.org.

[4] *The Globus Toolkit*, http://www-unix.globus.org/toolkit/about.html.

[5] *Trusted computer systems evaluation criteria (tcsec)*, no. US DOD 5200.28-STD, Department of Defence, United States, 1985.

[6] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, A. Gianoli, K. Lörentey, and F. Spataro, *VOMS, an Authorization System for Virtual Organizations*, Proceedings of the 1st European Across Grids Conference, Santiago de Compostela, Spain, February 2003, LNCS **2970** (2004), 33–40.

[7] American National Standard for Information Technology, *Role Based Access Control*, 2004, ANSI INCITS 359-2004 (approved 19 Feb 04).

[8] D. Bell and L. LaPadula, *Secure Computer System: Mathematical Foundations and Model*, Mitre technical report 2547, Mitre Corparation, Bedford, MA, 1975.

[9] D. Bell and L. LaPadula, *The Bell-LaPadula Model*, Journal of Computer Security **4** (1996), no. 2,3, 239–263.

[10] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, *The KeyNote Trust-Management System Version 2, IETF RFC 2704*, 1999.

[11] M. Blaze, J. Feigenbaum, and J. Lacy, *Decentralized Trust Management*, Proceedings of the 17th IEEE Symposium on Security and Privacy, IEEE Computer Society, 1996, pp. 164–173.

[12] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch, *A National-Scale Authentication Infrastructure*, IEEE Computer **33** (2000), no. 12, 60–66.

[13] T. Dierks and C. Allen, *The TLS Protocol, Version 1.0, IETF RFC 2246*, 1999.

[14] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, *SPKI Certificate Theory, IETF RFC 2693*, 1999.

[15] S. Farrell and R. Housley, *An Internet Attribute Certificate Profile for Authorization, IETF RFC 3281*, 2002.

[16] D.F. Ferraiolo and D.R. Kuhn, *Role Based Access Control*, Proceedings of the 15th National Computer Security Conference, 1992.

[17] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, ch. Globus: A Toolkit-Based Grid Architecture, pp. 259–278, Morgan Kaufmann, 1999.

[18] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, *A Security Architecture for Computational Grids*, Proceedings of the 5th ACM Conference on Computer and Communications Security Conference (CCS5), San Francisco, CA, USA, 1998, pp. 83–92.

[19] I. Foster, C. Kesselman, and S. Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, vol. 15, 2001, pp. 200–222.

[20] R. Housley, W. Ford, W. Polk, and D. Solo, *Internet X.509 Public Key Infrastructure Certificate and CRL Profile, IETF RFC 2459*, 1999, RFC459.

[21] International Telecommunication Union, *ITU-T Recommendation X.509, the Directory: Authentication Framework*, 2001, ISO/IEC 9594-8.

[22] J. B. D. Joshi, E. Bertino, U. Latif, and A. Ghafoor, *Generalized Temporal Role Based Access Control model (GTRBAC), Part I, Specification and Modeling*, Tech. Report 47, CERIAS, 2001.

[23] B. Lampson, *Protection*, Proceedings of the 5th Princeton Symposium on Information Sciences and Systems, Princeton, NJ, USA, 1971, pp. 437–443.

[24] J. Linn, *Generic Security Service Application Program Interface, Version 2, IETF RFC 2078*, 1997.

[25] M. Lorch, D.B. Adams, D. Kafura, M.S.R. Koneni, A. Rathi, and S. Shah, *The PRIMA System for Privilege Management, Authorization and Enforcement in Grid Environments*, Proceedings of the 4th International Workshop on Grid Computing (Grid2003), Phoenix, AZ, USA, Nov 2003.

[26] M. Lorch, B. Cowles, R. Baker, L. Gommans, P. Madsen, A. McNab, L. Ramakrishnan, K. Sankar, D. Skow, and M. R. Thompson, *Conceptual Grid Authorization Framework and Classification*, GGF Draft, Apr 2004, GWD-I.

[27] Zs. Németh and V. Sunderam, *Characterizing Grids: Attributes, Definitions, and Formalisms*, Journal of Grid Computing **1** (2003), no. 1, 9–23.

[28] L. Pearlman, C. Kesselman, Welch V., Foster I., and S. Tuecke, *The Community Authorization Service: Status and Future*, Proceedings of the Conference on Computing in High Energy and Nuclear Physics (CHEP 2003), La Jolla, CA, USA, 2003.

[29] B. Travica, *The Design of the Virtual Organization: a Research Model*, Proceedings of the Americas Conference on Information Systems (AMCIS 1997), Indianapolis, IN, USA, Aug 1997.

[30] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson, *Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, IETF RFC 3820*, Jun 2004.

[31] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist, *X.509 Proxy Certificates for dynamic delegation*, Proceedings of the 3rd Annual PKI R&D Workshop, Gaithersburg MD, USA, 2004.

[32] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke, *Security for Grid Services*, Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12), Seattle, WA, USA, 2003.

[33] P.R. Zimmermann, *The Official PGP User's Guide*, MIT Press, May 1995.