Role-based Secure Inter-operation and Resource Usage Management in Mobile Grid Systems

Antonios Gouglidis and Ioannis Mavridis

Department of Applied Informatics, University of Macedonia,
156 Egnatia Str., 54006, Thessaloniki, Greece.
{agougl,mavridis}@uom.gr

Abstract. Dynamic inter-domain collaborations and resource sharing comprise two key characteristics of mobile Grid systems. However, inter-domain collaborations have proven to be vulnerable to conflicts that can lead to privilege escalation. These conflicts are detectable in inter-operation policies, and occur due to cross-domain role relationships. In addition, resource sharing requires to be enhanced with resource usage management in virtual organizations where mobile nodes act as resource providers. In this case the enforcement of resource usage policies and quality of service policies are required to be supported due to the limited capabilities of the devices. Yet, the ANSI INCITS 359-2004 standard RBAC model provides neither any policy conflict resolution mechanism among domains, nor any resource usage management functionality. In this paper, we propose the domRBAC model for access control in mobile Grid systems at a low administrative overhead. The domRBAC is defined as an extension of the standardized RBAC by incorporating additional functionality to cope with requirements posed by the aforementioned systems. As a result, domRBAC facilitates collaborations among domains under secure inter-operation, and provides support for resource usage management in the context of multi-domain computing environments, where mobile nodes operate as first-class entities.

Keywords: mobile Grid, role based access control (RBAC), secure inter-operation, resource usage management, cross-domain authorization

1 Introduction

In recent years, Grid computing has become the focal point of science and enterprise computer environments. The Grid is an emergent technology that can be defined as a system able to share resources and provide problem solving in a coordinated manner within dynamic, multi-institutional virtual organizations [9]. This definition depends mostly on the sharing of resources and the collaboration of individual users or groups within the same or among different virtual organizations, in a service oriented approach. In turn, mobile Grid systems incorporate additional complexity and new challenges, due to the support of dynamic virtual organizations and the commercialization of Grid services [25]. Access control, in

such computing systems, is an active research area given the challenges and complex applications. The role of access control is to control and limit the actions or operations in a system performed by a user on a set of resources. In brief, it enforces the access control policy of a system and it prevents the access policy from subversion [3]. An extensive research has been done in the area of access control in collaborative systems [24], [28]. Nonetheless, further examination is demanded. This is mainly due to the partial or weak fulfilment of access control requirements in the aforementioned systems [11].

In this paper, we propose a new access control model called domRBAC to provide secure inter-operation among domains and resource usage management in collaborative systems, as the mobile Grid computing paradigm, where mobile devices can participate as first-class entities. Specifically we examine an incremental integration of individual RBAC policies into a global policy, which is suitable for dynamic virtual organizations. This is achieved via the definition of cross-domain mappings between roles. Thus, any user authorized for a role $d_i r_m$ in a domain d_i is granted access to all the permissions of its mapped role $d_j r_n$ in domain d_j. Nevertheless, inter-domain collaborations is a challenging task since they can lead to various types of conflict. Research in [10] has shown that secure inter-operation in federated systems must conform to the principles of autonomy and security. The principle of autonomy states that any access permitted within an individual system must also be permitted under secure inter-operation. Regarding the principle of security, it states that any access not permitted within an individual system must also be denied under secure inter-operation [10]. The former principles can be preserved in a collaboration, if a number of violations are successfully identified. Violations in role-based approaches, possibly leading to privilege escalation, can occur due to conflicts in cyclic inheritance and in static and dynamic separation of duty relations. In regard to resource usage management, domRBAC provides the capability of applying usage policies and, thus, enforcing quality of service rules on sharable resources. The application of resource usage policies can greatly amplify the adoption of Grid systems. For instance, it can be applied in Grid systems where ad-hoc mobile devices operate as first-class entities, or in multi-tenant environments, where usage based pricing is required.

The structure of the remainder of this paper is as follows. Section 2 provides information on related work and presents our motivation. Section 3 discuss domRBAC model in a systematic manner. A demonstration of the proposed model is given in section 4. Finally, we present our concluding remarks in section 5.

2 Relevant Work and Motivation

The access control models implemented by the existing Grid authorization mechanisms are either role based or attribute based. Role based access control (RBAC) approaches have gained considerable attention among researchers, due to ease of administration and support of a significant number of principles, namely the least privilege, separation of administrative functions and separation of duty re-

lationships [20]. However, RBAC handles better centralized architectures and is rather weak in inter-domain collaborations. Such functionality is absent from the ANSI INCITS 359-2004 [2]. Attribute based access control (ABAC) approaches have lately gained a lot of attention due to the development of internet based distributed systems. ABAC can provide access decisions on resources based on the requestor's owned attributes. A basic advantage of ABAC in comparison to RBAC is that in the former approach it is possible to provide access to users in a collaborative environment without the need for them to be known by the resource a priori. The UCON$_{ABC}$ model [15], [19] is a representative ABAC model, based on a modern conceptual framework, which encompasses traditional access control, trust management and digital rights management for the protection of digital resources.

Through time, numerous RBAC-based and ABAC-based models have been proposed trying to overcome some of the limitation of their initial implementations. In regard to RBAC and secure inter-operation, research in [21] proposed an integer programming (IP)-based approach for optimal resolution of the examined conflicts. A policy integration framework is used for the merging of the individual RBAC policies into a global policy. However, this approach is not dynamic, since the global policy is not a result of an incremental composition of the inter-domain policies. In [6] an inter-domain role-mapping approach based on the least privilege principle is suggested. Yet, the applied greedy algorithm may not compute optimal solutions, and from a security perspective may fail to find a safe solution. Research in [22] presents a protocol for secure inter-operation, which is based on the idea of access paths and access paths constraints. Nonetheless, the protocol does not check for violations during an inter-domain role assignment. Rather, it assumes that inter-domain role mappings already exist. In [26] the DRBAC is presented as a dynamic context-aware access control model for Grid applications. However, the management of inter-domain policies is not tackled. Resource usage management, to the best of our knowledge, is completely absent from the existing RBAC-based models. On the contrary, usage control was a subject of research in the UCON conceptual framework [15], [19], that is an ABAC model with the capability of enforcing RBAC policies. Nevertheless, UCON lacks administrative models and requires synchronized attribute acquisition and management that makes it more complex when applied to large systems.

In Grid systems, the existence of various access control models, inevitably led to the implementation of different Grid authorization mechanisms. Additionally, each mechanism tried to further implement features not intrinsically supported by the implemented model (i.e. support of inter-domain collaborations, quality of service and so on). Representative authorization mechanisms in Grid systems are the Community Authorization Service (CAS) [16], the Virtual Organization Membership Service (VOMS) [1], Akenti [23], PERMIS [4], [5], and Usage Based Authorization [27]. Regarding mobile Grid systems various architectures have been proposed to provide solutions, as the virtual cluster approach in [17], the mobile OGSI.NET [7] and the Akogrimo project [18]. Yet, the proposed autho-

rization mechanisms are complementary to existing Grid authorization services, as the A4C infrastructure in Akogrimo.

So far we have outlined the key requirements of mobile Grid systems for access control operation and administration, which are explicitly identified by the need for support of dynamic and secure inter-operation and interaction among the participating entities. To this extent, the examined access control models do not provide a solid solution to cope with these requirements of mobile Grid systems. Nonetheless, they are mostly targeted to general-purpose collaborative systems. Furthermore, the support of resource usage policies, in order to tackle the enforcement of quality of service policies is absent from RBAC family of models and only supported by the UCON conceptual framework. However, as discussed, ABAC solutions are prone to complexity when applied to large systems and lack administrative models. Therefore, we propose the domRBAC model, which combines the virtues of RBAC-based models and provides in addition resource usage management functionality in order to support modern Grid systems, as described in detail in the next section.

3 The Proposed domRBAC Model for Modern Collaborative Systems

The domRBAC model is an access control model capable of enforcing restrictive access control policies in collaborative systems, as the mobile Grid computing paradigm. The domRBAC model is based on the ANSI INCITS 359-2004 [2]. Thus, it supports all the components of the RBAC model, namely the core RBAC, hierarchical RBAC, static separation of duty relations, and dynamic of duty relations. However, domRBAC is enriched with additional functionality to cope with the requirements posed by modern computing environments. In this section, we discuss domRBAC model in a systematic manner.

3.1 domRBAC Elements

The domRBAC model consists of the following six basic elements: users, roles, sessions, operations, objects, and containers. Furthermore, domRBAC can support access control among domains. A domain can be defined as a protected computer environment, consisted of users and resources under an access control policy. This is done to cope with the problem of governing inter-operations among domains. Figure 1 illustrates the proposed access control model.

Sessions, objects and operations are three concepts that are commonly used in access control. The latter two form a new element of permissions. A permission or a privilege is an approval to perform an operation on one or more RBAC protected objects. In domRBAC, the aforementioned elements provide the same functionality in their familiar sense. As in all role-based models, sessions are dynamic elements. They are used as intermediary entities between the users and roles elements. The user element usually depicts a physical person who interfaces with a computer system. User elements, in role-based models, are

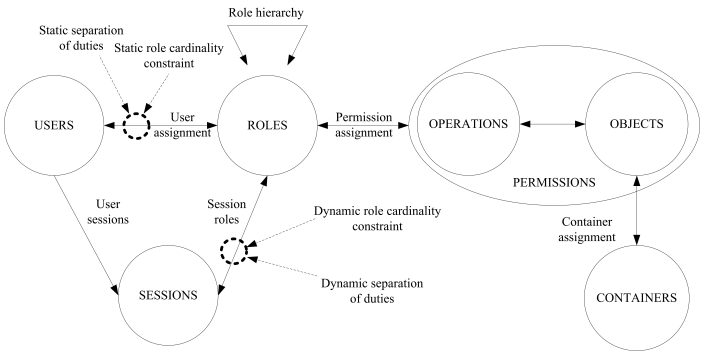

Fig. 1. The domRBAC access control model.

assigned to role elements and vice-versa. Sessions, in role-based models, are used to enforce dynamic security policies to computing systems. Each user can be associated with many sessions, and each session may have a combination of many active roles. Regarding objects, they are used to represent an entity in a computing system. Control of access to objects can be coarse-grained or fine-grained, depending on the computing system. For instance, the sharing of files and exhaustible system resources can be considered as an example of course-grained access control. On the contrary, the granting of access in a database on the level of record or field is an example of fine-grained access control. Yet, in domRBAC, an object can be associated with many container elements. The container element is explained in detail later in this section. Lastly, the element of operations provides a set of allowed operations on objects. Both operations and objects are system dependent. This means that different type of operations applies to different objects.

Roles in domRBAC are enriched with the notion of domains, and are expressed in pairs of domains and roles. For the naming of the roles, we use the *DomainRole* notation. Thus, the *Domain* prefix indicates the role's domain name, and the *Role* suffix indicates the name of the role. A formal definition is given later in definition 1.ii. The naming notation is used only for the element of roles. Nonetheless, when assigning users or permissions to roles, it is understood that the former two are also bound by the role's domain name. Through the role's naming convention, the domRBAC model can distinguish the security policies enforced among the autonomous domains.

The container is an abstract element that incorporates additional decision factors employed by the access decision function. The container can handle both environment and usage level information. The environment attributes are used to set time constraints, spatial information and so on and so forth. Yet, the usage level attributes can limit the usage of shared resources. The information specified in the container element is based on [14]. Thus, a container attribute can represent a certain property of the environment or usage levels. A container function provides a mechanism to obtain the current value of a specific container

attribute. Lastly, a container condition is a predicate that compares the current value of a container attribute either with a predefined constant, or another container attribute of the same domain. A significant enhancement of domRBAC as compared to the ANSI INCITS 359-2004 is that the element of container can support resource usage policies.

Moreover, domRBAC can support additional constraints, namely static and dynamic role cardinality constraints, which can be applied to the process of role assignment and role activation, respectively. This means that the number of roles that can be assigned to and/or activated by the users of a system can be managed. The constraint of role cardinality is introduced to fulfil both requirements posed by the system administrators as well as resource owners. Administrators can use static role cardinality to limit the assignment of critical roles with users. Furthermore, dynamic role cardinality can be used for setting quality of service rules. Resource owners can manage the usage of their resources by limiting the number of users that utilizes them. Thus, it is feasible to create license agreements between users and resource owners. This leads users to receive high quality services in a computing system.

Furthermore, the domRBAC model supports the identification of inter-domain violations, in an automated way, to avoid privilege escalation. The inter-domain violations are caused due to new immediate inter-domain role inheritance relations. The supported violations are: cyclic inheritance, violation of static separation of duty relations in a domain, and violation of dynamic separation of duty relations in a domain. Formal definitions are given later in this section in the Z formal description language [12] as in the ANSI INCITS 359-2004.

3.2 domRBAC Definitions

Definition 1. The core domRBAC.
The formal definition of core domRBAC model is based on [2], and is extended as follows:

i. USERS, ROLES, OPS, OBS, CNTRS, stands for users, roles, operations, objects and containers, respectively.
ii. $d_{domain}r_{role} \in$ ROLES is a role expressed in a DomainRole format, where Domain denotes a domain name and Role denotes a role name. For example, if a role r_m belongs to a domain d_i, we write $d_i r_m$.
iii. UA $\subseteq$ USERS $\times$ ROLES, a many-to-many mapping user-to-role assignment relation.
iv. assigned_users($d_i r_m$:ROLES) $\rightarrow 2^{USERS}$, the mapping of role $d_i r_m$ onto a set of users.
Formal definition: assigned_users($d_i r_m$) = {u $\in$ USERS | (u,$d_i r_m$) $\in$ UA}.
v. PRMS = $2^{(OPS \times OBS)}$, the set of permissions.
vi. PA $\subseteq$ PRMS $\times$ ROLES, a many-to-many mapping permission-to-role assignment relation.
vii. assigned_permissions($d_i r_m$:ROLES) $\rightarrow 2^{PRMS}$, the mapping of role $d_i r_m$ onto a set of permissions.

Formal definition: assigned_permissions($d_i r_m$)={p ∈ PRMS | (p,$d_i r_m$) ∈ PA}.

viii. CA ⊆ CNTRS × OBS, a many-to-many mapping container-to-object assignment relation.

ix. assigned_containers(o: OBS) → 2^{CNTRS}, the mapping of object o onto a set of containers.
Formal definition: assigned_containers(o)={c ∈ CNTRS | (c,o) ∈ CA}.

x. Op(p: PRMS) → {$op \subseteq OPS$}, the permission to operation mapping, which gives the set of operations associated with permission p.

xi. Ob(p: PRMS) → {$ob \subseteq OBS$}, the permission to object mapping, which gives the set of objects associated with permission p.

xii. SESSIONS = the set of sessions.

xiii. session_user(s: SESSIONS) → USERS, the mapping of session s onto a corresponding user.

xiv. session_roles(s: SESSIONS) → 2^{ROLES}, the mapping of session s onto a set of roles.
Formal definition: session_roles(s) ⊆ {$d_i r_m$ ∈ ROLES | (session_user(s),$d_i r_m$) ∈ UA}.

xv. avail_session_perms(s: SESSIONS) → 2^{PRMS}, the permissions available to a user in a session = $\bigcup_{d_i r_m \in session\_roles(s)} assigned\_permissions(d_i r_m)$.

Definition 2. Hierarchical domRBAC.
The hierarchical domRBAC is defined to cope with inter-domain role inheritance relations, and is enriched with notations from the theory of graphs. The reason why we choose to use the latter type of notation is bilateral. Firstly, graphs help in the visualisation of inter-domain role inheritance relations. Secondly, adjacency matrixes make it easy to find sub-graphs and adjacency queries are fast. Henceforth, we use i and j to refer to domains, where $i = j$ if we refer to an intra-domain relation, and $i \stackrel{+}{=} j$ if we refer to inter-domain relations ($intra - domain \subseteq inter - domain$).

i. $RH \subseteq ROLES \times ROLES$ is a partial order on $ROLES$ called the inheritance relation, written as $\geq$, where $d_i r_m \geq d_j r_n$ only if all permissions of $d_j r_n$ are also permissions of $d_i r_m$, and all users of $d_i r_m$ are also users of $d_j r_n$, i.e., $d_i r_m \geq d_j r_n$.
$\Rightarrow authorized\_permissions(d_j r_n) \subseteq authorized\_permissions(d_i r_m)$.

ii. $authorized\_users_{(i,j)}(d_i r_m : ROLES) \rightarrow 2^{USERS}$, the mapping of role $d_i r_m$ onto a set of users on the presence of a role hierarchy.
Formal definition: $authorized\_users_{(i,j)}(d_i r_m) = \{u \in USERS | d_j r_n \geq d_i r_m, (u, d_j r_n) \in UA\}$.

iii. $authorized\_permissions_{(i,j)}(d_i r_m : ROLES) \rightarrow 2^{PRMS}$, the mapping of role $d_i r_m$ onto a set of permissions in the presence of a role hierarchy.
Formal definition: $authorized\_permissions_{(i,j)}(d_i r_m) = \{p \in PRMS | d_j r_n \geq d_i r_m, (p, d_j r_n) \in PA\}$.

iv. $G = (V, E)$ is the inter-domain role hierarchy directed graph, which consists of a finite, nonempty set of role vertices $V \subseteq ROLES$ and a set of edges E.

Each edge is an ordered pair $(d_i r_m, d_j r_n)$, $i \stackrel{+}{=} j$ of role vertices that indicates the following relation: $d_i r_m \geq d_j r_n$.

v. A path in a G graph is a sequence of edges $(d_i r_1, d_i r_2)$, $(d_i r_2, d_i r_3)$, ..., $(d_i r_{n-1}, d_i r_n)$. This path is from role vertex $d_i r_1$ to role vertex $d_i r_n$ and has length n-1. The path represents not immediate inheritance relation between role vertex $d_i r_1$ and $d_i r_n$.

vi. A_G is the adjacency matrix representation for graph $G = (V, E)$, which is a $|V| \times |V|$ matrix, where $A_G[d_i r_m, d_j r_n] = 1$ if there is an edge from role vertex $d_i r_m$ to role vertex $d_j r_n$, and $A_G[d_i r_m, d_j r_n] = 0$ otherwise.

vii. Given a directed graph $G = (V, E)$ with adjacency matrix A_G, we compute a boolean matrix T_G such that $T_G[d_i r_m, d_j r_n]$ is 1 if there is a path from $d_i r_m$ to $d_j r_n$ of length 1 or more, and 0 otherwise. We call T the transitive closure of the adjacency matrix. For the computation of the transitive closure, the algorithm in [13] can be used. The algorithm computes the transitive closure of G in $O(|V|^3)$ time and $O(|V|^2)$ space.

viii. An adjacency list representation for graph $G = (V, E)$ is an array L of $|V|$ lists, one for each role vertex in V. For each role vertex $d_i r_m$, there is a pointer $L_{d_i r_m}$ to a linked list containing all the role vertices that are adjacent to $d_i r_m$.

ix. $DESCENDANT\_ROLES_{d_i r_m} \subseteq ROLES$ is a set that contains the role vertices of adjacency list $L_{d_i r_m}$. Thus, $DESCENDANT\_ROLES_{d_i r_m}$ contains all the roles in the inter-domain collaboration that are immediate or not immediate descendant roles of a given role $d_i r_m$.

Definition 3. Constrained domRBAC.
Apart from the support of static and dynamic separation of duty constraints in each domain, domRBAC supports static and dynamic role cardinality constraints. Static role cardinality constraints can restrict the number of users assigned to a role, to a maximum number. Moreover, dynamic role cardinality constraints can restrict the number of users that activate a role, to a maximum number in all concurrent sessions. In the following, we redefine SSD and DSD in the presence of domains, and we define static and dynamic role cardinality.

i. **Static Separation of duty (SSD):** SSD $\subseteq (2^{ROLES} \times N)$ is a collection of pairs ($d_i rs$,n) in SSD, where each $d_i rs$ is a role set in a domain d_i, t a subset of roles in $d_i rs$, and n is a natural number $\geq$2, with the property that no user of domain d_i is assigned to n or more roles from the set $d_i rs$ in each ($d_i rs$,n) $\in$ SSD.
Formal definition:
$\forall (d_i rs, n) \in SSD, \forall t \subseteq d_i rs : |t| \geq n \Rightarrow \bigcap_{d_i r_m \in t} assigned\_users(d_i r_m) = \emptyset$.

ii. **SSD in the presence of a hierarchy:** In the presence of a role hierarchy SSD is redefined based on authorized users rather than assigned users as follows:
Formal definition:
$\forall (d_i rs, n) \in SSD, i = j, \forall t \subseteq d_i rs : |t| \geq n$
$\Rightarrow \bigcap_{d_i r_m \in t} authorized\_users_{(i,j)}(d_i r_m) = \emptyset$.

iii. **Dynamic Separation of Duty (DSD):** DSD $\subseteq (2^{ROLES} \times N)$ is a collection of pairs ($d_i rs$,n) in DSD, where each $d_i rs$ is a role set and n a natural number $\geq$2, with the property that no subject may activate n or more roles from the set $d_i rs$ in each dsd $\in$ DSD.
Formal definition:
$\forall d_i rs \in 2^{ROLES}, n \in \mathbb{N}, (d_i rs, n) \in DSD \Rightarrow n \geq 2.|d_i rs| \geq n$, and
$\forall s \in SESSIONS, \forall d_i rs \in 2^{ROLES}$,
$\forall role\_subset \in 2^{ROLES}, \forall n \in \mathbb{N}, (d_i rs, n) \in DSD$,
$role\_subset \subseteq d_i rs, role\_subset \subseteq session\_roles(s) \Rightarrow |role\_subset| < n$.

iv. **Static role cardinality (SRC):** If static role cardinality constraint is required for any role $d_i r_m$, then $d_i r_m$ cannot be assigned to more than a maximum number of users.
SRC $\subseteq (ROLES \times N)$ is a collection of pairs ($d_i r_m$, n) in static role cardinality, where $d_i r_m$ is a role r_m in a domain d_i and n is a natural number ≥ 0, with the property that the number of users assigned with role $d_i r_m$ cannot exceed the number n in each ($d_i r_m$, n) $\in$ SRC.
Formal definition:
$d_i r_m \in ROLES, n \in \mathbb{N}, n \geq 0$,
$\forall (d_i r_m, n) \in SRC \Rightarrow |assigned\_users(d_i r_m)| \leq n$.

v. **SRC in the presence of a hierarchy:** In the presence of a role hierarchy static role cardinality constraint is redefined based on authorized users rather than assigned users as follows:
$d_i r_m \in ROLES, i \stackrel{+}{=} j, n \in \mathbb{N}, n \geq 0$,
$\forall (d_i r_m, n) \in SRC \Rightarrow |authorized\_users_{(i,j)}(d_i r_m)| \leq n$.

vi. **Dynamic role cardinality constraint (DRC):** If dynamic role cardinality is required for any role $d_i r_m$, then $d_i r_m$ cannot be activated for more than a maximum number of authorized users in all concurrent sessions of a system.
DRC $\subseteq (ROLES \times N)$ is a collection of pairs ($d_i r_m$, n) in dynamic role cardinality, where $d_i r_m$ is a role r_m and n is a natural number ≥ 0, with the property that the number of concurrent role activations by users authorized for role $d_i r_m$ cannot exceed the number n.
Formal definition:
$d_i r_m \in ROLES, n \in \mathbb{N}, n \geq 0$,
$\forall s \in$ SESSIONS, ($d_i r_m$,n) $\in$ DRC $\Rightarrow \sum |d_i r_m \cap session\_roles(s)| \leq n$.

After defining both the container element and the DRC constraint, we elaborate on the supported types of resource usage policies. The first type is via the container element, by declaring the required attribute value, function and condition of the container. However, this type of resource usage policy is unable to provide quality of service to consumers, since each container element restricts the usage of a resource on per role activation. A second type of resource usage policy with quality of service capabilities is provided via the combination of the container element and DRC constraint. This type of resource usage policy enforcement restricts the usage of a resource on all concurrent role activations.

Definition 4. Role Inheritance Management in domRBAC.
The domRBAC model aims at providing a comprehensive solution to secure inter-operation based on the principles of autonomy, security and containment. In order to establish a secure inter-operation among the participating domains, domRBAC provides two new administrative commands for managing inter-domain role inheritance relations. The administrative commands can be used by the administrator of each domain, according to the inter-operability requirements of each system. Their objective is to check for a number of violations before committing an inter-domain role inheritance relation. Thus, based on the definitions 4.i, 4.ii and 4.iii, we introduce the *InterdomainPolicyViolation* function for the checking of inter-domain violations due to the inter-domain role inheritance relations, and two new inter-domain administrative commands *AddInterdomainInheritance* and *DeleteInterdomainInheritance* for establishing and discarding immediate inter-domain inheritance relationships, respectively. Our approach utilizes algorithms derived from the theory of graphs. Intra-domain management not listed below is handled the same as in the ANSI INCITS 359-2004.

i. **Inter-domain violation of role assignment:** As stated in [21] an inter-domain policy causes a violation of role assignment constraint of domain d_i if it is allowed to a user u of domain d_i to access a local role $d_i r_m$ even though u is not directly assigned to $d_i r_m$ or any of the roles that are senior to $d_i r_m$ in the role hierarchy of domain d_i.
We identify role assignment violations by checking for cyclic inheritance in the inter-domain role hierarchy graph. Role assignment violations can occur due to the addition of a new immediate inter-domain inheritance relationship $d_i r_{m_{asc}} \gg d_j r_{n_{desc}}$ between existing roles $d_i r_{m_{asc}}$, $d_j r_{n_{desc}}$, where $d_i r_{m_{asc}}$ is a role ascendant of $d_j r_{n_{desc}}$.
The algorithm for detecting inter-domain violations of role assignment is given in Table 1.

Table 1. Inter-domain violation of role assignment algorithm

```
1.  ci_violation(d_i r_{m_asc}, d_j r_{n_desc}) : boolean
2.      for each d_i r ∈ DESCENDANT_ROLES_{d_i r_{m_asc}}
3.          for each d_j r ∈ DESCENDANT_ROLES_{d_j r_{n_desc}}
4.              if not ((T_G[d_i r_{m_asc}, d_i r] = 0 or
5.                  (T_G[d_i r_{m_asc}, d_i r] = 1 and d_i r_{m_asc} ≥ d_i r)) and
6.                  (T_G[d_j r_{n_desc}, d_j r] = 0 or
7.                  (T_G[d_j r_{n_desc}, d_j r] = 1 and d_j r_{n_desc} ≥ d_j r)))
8.              then
9.                  return true
10.     return false
```

ii. **Intra-domain violation of SSD relationships:** An inter-domain policy causes an intra-domain violation of SSD relationships of domain d_i if it is

allowed to a user u of domain d_i to be assigned to any two conflicting roles $d_i r_m$ and $d_i r_n$ of domain d_i. We identify violations of SSD relationships, using the following properties [8]:
Property 1: If there are two roles $d_i r_m$ and $d_j r_n$ that are mutually exclusive, then neither one should inherit the other, either directly or indirectly.
Property 2: If there are two roles $d_i r_m$ and $d_j r_n$ that are mutually exclusive, then there can be no third role that inherits both of them.
The algorithm for detecting intra-domain violations of SSD relationships is given in Table 2. Specifically, *Property 1* is maintained in lines 6-7, and *Property 2* in lines 8-9.

Table 2. Intra-domain violation of SSD relationships

```
1.  ssd_violation(d_i r_{m_asc}, d_j r_{n_desc}) : boolean
2.      d_i r ∈ DESCENDANT_ROLES_{d_i r_{m_asc}}
3.      d_j r ∈ DESCENDANT_ROLES_{d_j r_{n_desc}}
4.      for each (d_i r_m, d_i r_n) ∈ SSD_{d_i}
5.          for each (d_j r_m, d_j r_n) ∈ SSD_{d_j}
6.              if not (T_G[d_i r_m, d_i r_n] = 0 and T_G[d_i r_n, d_i r_m] = 0 and
7.                  T_G[d_j r_m, d_j r_n] = 0 and T_G[d_j r_n, d_j r_m] = 0 and
8.                  T_G[d_i r, d_i r_m] = 0 and T_G[d_i r, d_i r_n] = 0 and
9.                  T_G[d_j r, d_j r_m] = 0 and T_G[d_j r, d_j r_n] = 0)
10.             then
11.                 return true
12.     return false
```

iii. **Intra-domain violation of DSD relationships:** An inter-domain policy causes an intra-domain violation of DSD relationships of domain d_i if it is allowed to a user u of domain d_i to activate any two conflicting roles $d_i r_m$ and $d_i r_n$ of domain d_i. We identify violations of DSD relationships similarly to definition 4.ii due to the following property [8]:
Property 3: If SSD holds, then DSD is maintained. Thus, properties 1 and 2 must be guaranteed.
The algorithm for detecting intra-domain violations of DSD relationships is given in Table 3.

iv. **InterdomainPolicyViolation:** This function checks if violations 4.i, 4.ii and 4.iii occur during an inter-domain role inheritance relation. It returns **true** if a violation occurs from an inter-domain role association, and **false** otherwise. Table 4 presents the implementation of the function.

v. **AddInterdomainInheritance:** This command establishes a new immediate inter-domain inheritance relationship $d_i r_{m_{asc}} \gg d_j r_{n_{desc}}$ between existing roles $d_i r_{m_{asc}}$, $d_j r_{n_{desc}}$. The command is valid if and only if $d_i r_{m_{asc}}$ and $d_j r_{n_{desc}}$ are members of the $ROLES$ dataset, $d_i r_{m_{asc}}$ is not an immediate ascendant of $d_j r_{n_{desc}}$, and violations of role assignment and of SSD and DSD relationships do not occur.

Table 3. Intra-domain violation of DSD relationships

1. **dsd_violation**$(d_ir_{m_{asc}}, d_jr_{n_{desc}})$: **boolean**
2. $d_ir \in DESCENDANT\_ROLES_{d_ir_{m_{asc}}}$
3. $d_jr \in DESCENDANT\_ROLES_{d_jr_{n_{desc}}}$
4. **for each** $(d_ir_m, d_ir_n) \in DSD_{d_i}$
5. **for each** $(d_jr_m, d_jr_n) \in DSD_{d_j}$
6. **if not** $(T_G[d_ir_m, d_ir_n] = 0$ **and** $T_G[d_ir_n, d_ir_m] = 0$ **and**
7. $T_G[d_jr_m, d_jr_n] = 0$ **and** $T_G[d_jr_n, d_jr_m] = 0$ **and**
8. $T_G[d_ir, d_ir_m] = 0$ **and** $T_G[d_ir, d_ir_n] = 0$ **and**
9. $T_G[d_jr, d_jr_m] = 0$ **and** $T_G[d_jr, d_jr_n] = 0)$
10. **then**
11. **return true**
12. **return false**

Table 4. Inter-domain policy violation function

1. **InterdomainPolicyViolation**$(d_ir_{m_{asc}}, d_jr_{n_{desc}})$: **boolean**
2. **return ci_violation**$(d_ir_{m_{asc}}, d_jr_{n_{desc}})$ **or**
3. **ssd_violation**$(d_ir_{m_{asc}}, d_jr_{n_{desc}})$ **or**
4. **dsd_violation**$(d_ir_{m_{asc}}, d_jr_{n_{desc}})$

Formal definition:
$AddInterdomainInheritance(d_ir_{m_{asc}}, d_jr_{n_{desc}} : NAME) \lhd$
$d_ir_{m_{asc}}, d_jr_{n_{desc}} \in ROLES;$
$InterdomainPolicyViolation(d_ir_{m_{asc}}, d_jr_{n_{desc}}) = false;$
$\neg(d_ir_{m_{asc}} \gg d_jr_{n_{desc}}); \neg(d_jr_{n_{desc}} \geq d_ir_{m_{asc}})$
$\geq' = \geq \cup \{dr, dq : ROLES | dr \geq d_ir_{m_{asc}} \wedge d_jr_{n_{desc}} \geq dq \bullet dr \mapsto dq\} \rhd$

vi. **DeleteInterdomainInheritance:** This command deletes an existing immediate inter-domain inheritance relationship $d_ir_{m_{asc}} \gg d_jr_{n_{desc}}$. The command is valid if and only if the roles $d_ir_{m_{asc}}$ and $d_jr_{n_{desc}}$ are members of the $ROLES$ dataset, and $d_ir_{m_{asc}}$ is an immediate ascendant of $d_jr_{n_{desc}}$. The new inter-domain inheritance relation is computed as the reflexive-transitive closure of the immediate inheritance relation resulted after deleting the relationship $d_ir_{m_{asc}} \gg d_jr_{n_{desc}}$.
Formal definition:
$DeleteInterdomainInheritance(d_ir_{m_{asc}}, d_jr_{n_{desc}} : NAME) \lhd$
$d_ir_{m_{asc}}, d_jr_{n_{desc}} \in ROLES; (d_ir_{m_{asc}} \gg d_jr_{n_{desc}})$
$\geq' = (\gg \ \{d_ir_{m_{asc}} \mapsto d_jr_{n_{desc}}\})^* \rhd$

4 Use Cases

In this section, we describe two contrived use cases to demonstrate the newly introduced functionality of domRBAC. The first use case demonstrates how to

enforce resource usage policies, and the second how to identify security violations in an inter-domain role inheritance relation.

4.1 Use Case 1: Resource Usage Management

Figure 2(a) shows a simple policy in a domain d_1. Role d_1r_a is a role senior to d_1r_b. User Alice requires to share the CPU cycles of her mobile device. Since the CPU capabilities of the device are limited, she decides to share only 50% of her CPU cycles, and to provide to each consumer at most 5% of her sharable CPU cycles. In order to apply the aforementioned policy, role d_1r_b is assigned to permission $P_{UC} = (Usage, CPU)$. This means that a usage operation is assigned to a CPU object. A container c_B is assigned to object CPU. Container c_B has the following properties: a container attribute that defines the CPU usage value equal to 5%, a container function that returns the current CPU usage, and a container condition $\leq$. Moreover, a DRC constraint is applied to limit the number of active users to 10 ($DRC_{d_1r_b} = (d_1r_b, 10)$). The latter constraint assures that the number of concurrent active users cannot exceed the 10 users. Thus, in conjunction with the container element it is assured that the usage of CPU not exceed the 50%, and that each consumer receive at most 5% of CPU. If the DRC constraint was omitted, Alice would not be able to limit the usage of her resources, nor guarantee 5% of CPU usage to the consumers.

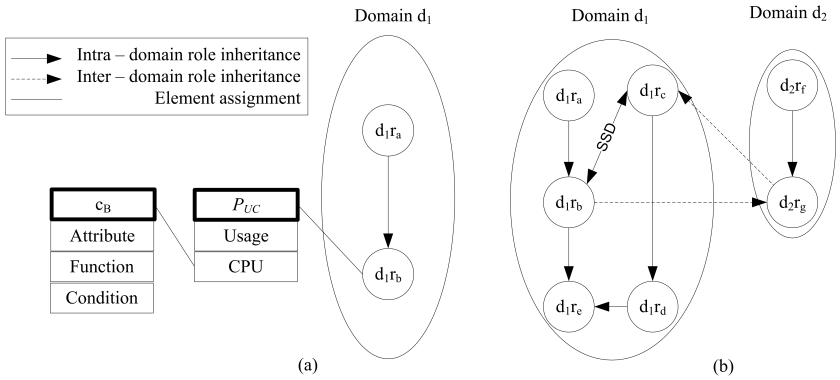

Fig. 2. (a) Resource usage management use case. (b) Security violation use case.

4.2 Use Case 2: Security Violation

Figure 2(b) shows a multi-domain policy that allows collaboration between domain d_1 and domain d_2. Domain d_1 has the following roles: d_1r_a, d_1r_b, d_1r_c, d_1r_d and d_1r_e. Role d_1r_a inherits all permissions of d_1r_b which further inherits d_1r_e. Role d_1r_c inherits all permissions of d_1r_d which further inherits d_1r_e. A

static separation of duty relation is specified for d_1r_b and d_1r_c meaning that these roles cannot be assigned to the same user simultaneously. Domain d_2 has the following roles: d_2r_f and d_2r_g. Role d_2r_f inherits all permissions of d_2r_g. The policy defines the following inter-operation between domains d_1 and d_2.

i. Role d_1r_b inherits all the permissions available to role d_2r_g.
ii. Role d_2r_g inherits all the permissions available to role d_1r_c.

However, the multi-domain policy leads to a violation of a SSD relationship in domain d_1. It allows d_1r_b to access the permissions of role d_1r_c through d_2r_g. Policy i does not raise any of the discussed violations. Regarding policy ii we work as follows:
Step 1. We assume that policy ii can be enforced.
Step 2. We construct the adjacency matrix A_G, which contains the inter-domain role hierarchy and we compute the sets of descendant roles for the two roles used in the multi-domain policy.

$$A_G = \begin{array}{c} \\ d_1r_a \\ d_1r_b \\ d_1r_c \\ d_1r_d \\ d_1r_e \\ d_2r_f \\ d_2r_g \end{array} \begin{array}{c} \begin{array}{ccccccc} d_1r_a & d_1r_b & d_1r_c & d_1r_d & d_1r_e & d_2r_f & d_2r_g \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$DESCENDANT\_ROLES_{d_2r_g} = \{d_1r_c, d_1r_d, d_1r_e\}$, and
$DESCENDANT\_ROLES_{d_1r_c} = \{d_1r_d, d_1r_e\}$.
Step 3. We call function $InterdomainPolicyViolation$ with function parameters d_2r_g and d_1r_c, and we compute the $ssd\_violation(d_2r_g, d_1r_c)$. Based on definition 4.ii: $T_G[d_1r_b, d_1r_c] = 1 \Rightarrow ssd\_violation(d_2r_g, d_1r_c) = true$. The identification of a inter-domain violation of SSD relation will discard the inter-domain inheritance relationship, assumed in the hypothesis of step 1.

5 Conclusion

The proposed domRBAC model is an extended RBAC model with enhancements stemmed from a list of requirements from mobile Grid systems and commercialized applications. The applied enhancements result in a robust, scalable and dynamic access control model. The domRBAC model takes advantage of all the virtues of RBAC family of models, and additionally, encompasses features from ABAC approaches such as multi-domain support and resource usage management. Opposed to existing solutions, secure inter-operation among domains in domRBAC is achieved by checking gradually and dynamically for violations in inter-domain role inheritance relations, as required by mobile Grid systems. Furthermore, resource usage management is firstly introduced in an RBAC-based

access control model, as a requirement for enforcing quality of service policies. Future work includes the design of an architecture that will implement the proposed access control model and a performance study.

References

1. Alfieri, R., Cecchini, R., Ciaschini, V., Dell' Agnello, L., Frohner, A., Gianoli, A., Lorentey, K., Spataro, F.: Voms, an authorization system for virtual organizations. In: Grid Computing. pp. 33–40. Springer (2004)
2. American National Standard Institute, I.: Ansi incits 359-2004, role based access control (2004)
3. Benantar, M.: Access Control Systems: Security, Identity Management and Trust Models. Springer-Verlag New York, Inc. (2005)
4. Chadwick, D.: Authorisation in grid computing. Information Security Technical Report 10(1), 33–40 (2005)
5. Chadwick, D., Otenko, A., Ball, E.: Role-based access control with x. 509 attribute certificates. Internet Computing, IEEE 7(2), 62–69 (2003)
6. Chen, L., Crampton, J.: Inter-domain role mapping and least privilege. In: SACMAT '07: Proceedings of the 12th ACM symposium on Access control models and technologies. pp. 157–162. ACM, New York, NY, USA (2007)
7. Chu, D.C., Humphrey, M.: Mobile ogsi.net: Grid computing on mobile devices. Grid Computing, IEEE/ACM International Workshop on 0, 182–191 (2004)
8. Ferraiolo, D.F., Kuhn, D.R., Chandramouli, R.: Role-Based Access Control. Artech House, Inc. (2003)
9. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. International Journal of High Performance Computing Applications 15(3), 200 (2001)
10. Gong, L., Qian, X.: Computational issues in secure interoperation. IEEE Trans. Softw. Eng. 22(1), 43–52 (1996)
11. Gouglidis, A., Mavridis, I.: On the definition of access control requirements for grid and cloud computing systems. In: Networks for Grid Applications, LNICST, vol. 25, pp. 19–26. Springer Berlin Heidelberg (2010)
12. ISO/IEC-13568: Information technology z - formal specification notation - syntax, type system and semantics (2002), international Standard
13. Jonathan L. Gross, J.Y. (ed.): Handbook of Graph Theory (Discrete Mathematics and Its Applications). CRC, 1 edn. (December 2003)
14. Neumann, G., Strembeck, M.: An approach to engineer and enforce context constraints in an rbac environment. In: SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies. pp. 65–79. ACM, New York, NY, USA (2003)
15. Park, J., Sandhu, R.: The ucon abc usage control model. ACM Trans. Inf. Syst. Secur. 7(1), 128–174 (2004)
16. Pearlman, L., Welch, V., Foster, I., Kesselman, C., Tuecke, S.: A community authorization service for group collaboration. In: Policies for Distributed Systems and Networks. Proceedings. Third International Workshop on. pp. 50–59. IEEE (2002)
17. Phan, T., Huang, L., Dulan, C.: Challenge: integrating mobile wireless devices into the computational grid. In: Proceedings of the 8th annual international conference on Mobile computing and networking. p. 278. ACM (2002)

18. Racz, P., Burgos, J., Inacio, N., Morariu, C., Olmedo, V., Villagra, V., Aguiar, R., Stiller, B.: Mobility and qos support for a commercial mobile grid in akogrimo. In: Mobile and Wireless Communications Summit, 2007. 16th IST. pp. 1 –5 (2007)
19. Sandhu, R., Park, J.: Usage control: A vision for next generation access control. In: Computer Network Security, vol. 2776, pp. 17–31. Springer Berlin / Heidelberg (2003)
20. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
21. Shafiq, B., Joshi, J.B.D., Bertino, E., Ghafoor, A.: Secure interoperation in a multidomain environment employing rbac policies. IEEE Trans. on Knowl. and Data Eng. 17(11), 1557–1577 (2005)
22. Shehab, M., Bertino, E., Ghafoor, A.: Serat: Secure role mapping technique for decentralized secure interoperability. In: SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies. pp. 159–167. ACM, New York, NY, USA (2005)
23. Thompson, M., Essiari, A., Mudumbai, S.: Certificate-based authorization policy in a pki environment. ACM Transactions on Information and System Security (TISSEC) 6(4), 566–588 (2003)
24. Tolone, W., Ahn, G.J., Pai, T., Hong, S.P.: Access control in collaborative systems. ACM Comput. Surv. 37(1), 29–41 (2005)
25. Waldburger, M., Stiller, B.: Regulatory issues for mobile grid computing in the european union. In: 17th European Regional ITS Conference, Amsterdam, The Netherlands. pp. 1–9 (2006)
26. Zhang, G., Parashar, M.: Dynamic context-aware access control for grid applications. In: Grid Computing, 2003. Proceedings. Fourth International Workshop on. pp. 101–108. IEEE (2004)
27. Zhang, X., Nakae, M., Covington, M., Sandhu, R.: A usage-based authorization framework for collaborative computing systems. In: Proceedings of the 11th ACM symposium on Access control models and technologies. pp. 180–189. ACM (2006)
28. Zhang, X., Nakae, M., Covington, M.J., Sandhu, R.: Toward a usage-based security framework for collaborative computing systems. ACM Trans. Inf. Syst. Secur. 11(1), 1–36 (2008)