Hierarchical domains for decentralized administration of spatially-aware RBAC systems

Maria Luisa Damiani
University of Milan, Italy
EPFL, Switzerland

Claudio Silvestri
University of Milan, Italy

Elisa Bertino
Purdue University, US

Abstract

Emerging models for context-aware role-based access control pose challenging requirements over policy administration. In this paper we address the issues raised by the decentralized administration of a spatially-aware access control model in a mobile setting. We present in particular GEO-RBAC Admin, the administration model for the GEO-RBAC model. The model is grounded on the notion of hierarchy of spatial domains where a spatial domain is an entity which collects objects based on organizational and spatial proximity criteria.

1 Introduction

The administration of RBAC (Role Based Access Control) policies in large organizations is an important research issue, which is gaining new momentum under the push of new emerging paradigms like context-aware access control models [1, 2, 3, 6, 13, 15] and parametrized roles [10, 11, 16].

Typically access control policies are administered in accordance with an *administrative model*. An administrative model defines a language for policy specification. For example, the administration model of the RBAC standard defines a set of functions comprising *administrative functions* for the creation and maintenance of the element sets and relations of the RBAC system, and *review functions* to review the results of the actions created by administrative actions. Permissions are considered predefined by the underlying information system in which RBAC is deployed. In most cases administration is *centralized*, that is, the policy is administered by a unique administrator [9].

Unfortunately RBAC, when applied to large organizations consisting of thousands of roles and users [20] does not scale well because it lacks mechanisms for the modular organization of roles and the decentralized management of policies. Note that we use the terms administration decentralization and administration delegation as synonyms to mean that *someone has the authority of assigning administrative functions to somebody else* and thus of "distributing" the administration tasks.

To overcome this limit of RBAC, an appealing approach is to extend the model with the notion of *administrative domain* (simply *domain* hereinafter). A domain basically denotes a portion of the overall policy which is administered by one or more autonomous administrators. The problem of how to extend RBAC with domains raises however important research issues which have been only partially addressed, such as how to represent domains.

Even more challenging is the introduction of this concept in a RBAC-based spatially-aware access control model. Broadly speaking, we say that access control is spatially-aware if the authorization to access depends on the position of the user. The growing interest for this class of models is motivated by the great concern for the security issues in mobile settings. In this paper we focus in particular on the decentralized administration of policies relying on the GEO-RBAC model [8]. Crucial issues include how to account for the spatial dimension in the administration; how to rigorously define domain and administration delegation; and how to account of the unusual complex structure of roles comprising attributes and role types.

The models proposed for the administration of RBAC systems are not able to address such issues. We have thus defined the *GEO-RBAC Admin* model. The key idea of the model is the concept of *hierarchy of domains*. A domain is a first class entity which is associated with a *owner role*: the owner role is the unique role having an authority over the domain. Domains can be dynamically and recursively decomposed in smaller domains which are administered by administration roles based on a rigorously defined mechanism of administration delegation. The resulting model which combines the concept of ownership typical of discretionary access control models with the concepts of domains and roles, supports a decentralized administration under the control of the trusted top administrator. Despite of the focus

on GEO-RBAC, we believe that the proposed approach is of more general applicability, and can be easily generalized to be used in different authorization administration models.

This paper is organized as follows: the next section outlines major requirements and how they have been addressed in related work; then the key design choices in *GEO-RBAC Admin* are illustrated; the model is then formally described in the subsequent section. Final remarks conclude the paper.

2 Motivations and requirements

We begin by briefly summarizing key concepts of GEO-RBAC. We then define an ontological view of domains, the requirements which follow and the related work.

2.1 GEO-RBAC

The model is based on the central concept of *spatial role*. A spatial role is a spatially confined organizational function. A spatial roles becomes effective only when the subject (the *user* in the RBAC terminology) who has been assigned that role is located in the *role extent*. In that case the role is said to be enabled. Both the role extent and the user positions are described as spatial objects (i.e. *spatial features* compliant with geo-spatial standards) of the type specified in the *role schema*.

The role schema is the template for spatial roles. For example the role schema for doctor, specified as *doctor(HospitalType, LocationType, Lmf)* is read as follows: *doctor* is the role name, *HospitalType* is the role extent type; *LocationType* is the type of *logical position* which describes the position of the user at a certain granularity, for example building and room; *Lmf* is the *location mapping function* which maps the actual position acquired through some location sensing technology to the logical and abstract location. We refer the reader to [8] for more details.

2.2 Ontological view of domains and requirements

The set of spatial entities which are defined in a GEO-RBAC policy, such as spatial roles, are assumed to have a geometric extent which falls inside a possibly bounded reference space. Such reference space defines the spatial scope of the organization.

The geographical location represents a natural element of cohesion inside a community or an organization. Therefore in large and distributed organizations, consisting of many units located in different regions, it seems reasonable to think of the reference space of the access control policy as a composite space consisting of diverse places and then relate each of these places to a domain. Accordingly we propose the following definition of GEO-RBAC domain which extends the one proposed by Kern & al. [14] to describe a generic domain: a GEO-RBAC domain is *an entity which collects objects according to some characteristic, such as the organizational structure*, **and spatial proximity**. We highlight the property of *spatial proximity* to point out that a domain represents some sector of an organization which can be associated with a location and in which objects and users are physically close to each other.

From this ontological perspective, it follows the requirement that the domain model must have a spatial connotation. Further we envision a scenario in which domains can be dynamically decomposed in smaller domains (*sub-domains*), so to enable a more flexible administration. The definition of the notion of sub-domain requires addressing various questions: which kind of relationship exists between a domain and its sub-domains? Which is the degree of autonomy of sub-domain administrators? To our knowledge these issues have been not addressed before. We also recall that the space dimension in GEO-RBAC is used to strengthen security, therefore the design of the administrative model must be compliant with this general goal. Finally, it is important to stress that the administrative model should be finalized to the development of a flexible and comfortable platform for policy specification and thus it is important to base the design of the model on principles like simplicity and clearness [16].

2.3 Related work

The spatial dimension of domains is a novel concept that has not been investigated before. Related work is thus about the decentralization of administration in classical models. To our knowledge, the first to propose the notion of domain in access control has been Moffet, in his PhD thesis entitled "Delegation of Authority Using Domain Based Access Rules" (1990) [17]. This work presents an extension of the access control matrix in which a domain denotes a collection of objects which are explicitly grouped together for a purpose. Moreover domains have administrators which can delegate to other users the administration of other domains. This idea, that resembles our approach, did not have, however, a strong follow-up at level of research.

The concern for the decentralized administration in RBAC systems presents many variations. In this section we overview significant approaches focusing, in particular, on the following two questions:

- How can a domain be represented in RBAC systems?
- Who can delegate the administration of what to whom?

How can a domain be represented?

In ARBAC97 [19] a domain is indirectly represented through the notion of *role range*, namely a pair of roles confining the portion of role hierarchy, namely the set of roles, that an administrator can administer. Role hierarchy characterizes also the administrative model proposed by Crampton [7]. Oh and Sandhu in [18] propose an administration model named ARBAC02. ARBAC02 retains the main features of ARBAC97, and adds the concept of organization unit, which represents a group of individual involved in related tasks that are modeled through the concepts of user pool and permission pool which seem of little generality. A different approach is proposed in UARBAC [16]. UARBAC is a family of administrative models for RBAC which consists of a basic model and one extension $UARBAC^P$. In particular $UARBAC^P$ adds constraint-based administrative domains: the basic idea is to assign one or more attributes to each object in the RBAC system and then define administrative domains using constraints on these attributes. The notion of domain is however not explicit, in that it can be represented by the value of some arbitrary attribute of objects or even by a combination of values of attributes. Further this approach makes strong assumptions on the nature of the RBAC model, in that domains can only be added if objects are parametrized.

The administration of RBAC-based context-aware access control has been addressed by X-GTRBAC Admin [5]. A salient feature of this model is that domains are first class objects, namely have an identifier and can be related to other objects, in much the same way an Internet domain is defined by a name. A policy is then explicitly associated with a domain. Despite its simplicity, this notion of domain is fairly powerful, since it is actually independent from the concepts of the underlying RBAC and thus can be naturally added to a variety of models.

Who can delegate the administration of what to whom?

A major mechanism for administration delegation is provided by discretionary access control (DAC) models [4, 12], in that a subject with a certain access permission is capable of passing that permission on to another subject. Typically the delegation is performed separately for each permission and subject. Further, there is no a-priori distinction between the subjects that can administer a permission and those who cannot. Moreover, it lacks the notion of domain as aggregation of subjects and permissions.

In the decentralized administration models based on RBAC, domains are administered by members of roles. The roles which are enabled to exercise administration functions may be either *administration roles* like in ARBAC97 and X-GTRBAC or application-dependent roles like in UARBAC. Separating administrative roles from non-administrative roles ensures a stronger security control and thus is preferable in contexts in which strong security is requested. In ARBAC97 and X-GTRBAC the Security System Officer is the only subject allowed to create administrative roles and thus decentralize administration. The delegation hierarchy thus consists of only two levels, the Security System Officer and the application-dependent administrative roles. In large organizations, however, a delegation hierarchy organized on two levels may be not sufficiently flexible because of the complexity of the organizational structure.

3 Baseline of the approach

Current approaches provide only partial solutions to the requirements posed by the the GEO-RBAC administration. We have thus investigated a model which tries to rigorously introduce the spatial dimension in the notion of domain. GEO-RBAC Admin is based on a number of key concepts: domain and domain hierarchy, delegation permission, domain scoping rules, soundness and administration completeness of domains.

3.1 The representation of domains

Following the approach proposed in X-GTRBAC Admin, we consider domains as first class objects because such an approach ensures a clean design. However, unlike X-GTRBAC, a domain, besides a name, has attributes. One attribute is mandatory, namely the *reference space* of the domain. The reference space of domain d is the (possibly bounded) space, which confines the spatial entities and the position of users defined in d. The reference space is represented by a spatial feature of system-defined type. *Top-Domain* is the system-defined domain; conventionally its reference space is the feature covering the whole Earth.

3.2 Administration of domains

A domain can be assigned one or more *admin roles* (administration roles) and *regular roles* where regular stands for non-administrative. Admin roles are assigned *admin users* while regular roles are assigned *regular users*. The set of admin roles (admin users) is disjoint from the set of regular roles (regular users).

An admin role is uniformly represented as a GEO-RBAC role [1]: in particular the admin role is a role defined over the extent representing the reference space of the domain.

[1] Specifically an admin role is a GEO-RBAC *non-spatial* role, because the role extent and the logical position coincide with the reference space and that matches with the definition of non-spatial role

Admin roles are instances of *admin schemas*. We emphasize that admin schemas are used in domains for a specific purpose, specifically to define templates for the admin roles in sub-domains. Such templates define a common name and a set of permissions which, for how the GEO-RBAC model is defined, are automatically assigned to all instances of the schema. For example one can define the role schema $AdminUser$, assign it a set of permissions and then create admin roles in sub-domains as instances of that schema, say $AdminUser(subd_1), AdminUser(subd_2)$, where $subd_1$ and $subd_2$ are reference spaces of sub-domains. This concept of admin schema ultimately enables the modular organization of administrative roles and permissions in sub-domains.

Conventionally *TopAdmin* is the unique admin role in the Top Domain and it has the whole set of permissions.

3.3 Who creates domains? The administration delegation issue

A first issue is to define who administers the sub-domain. The key choice is that the only role who has such authority is the admin role of the user who has created the sub-domain. Such a role is named *owner role*. Each sub-domain, except for Top Domain, has thus a *owner role*: it is the unique role that has an authority over the sub-domain and thus can create and confer power (and revoke) to administrators of sub-domains. Therefore it cannot occur that admin roles in a sub-domain are created by different roles; either it cannot occur that the admin role of a sub-domain is assigned permissions by different roles.

The next issue is how to authorize delegation. We say that an admin role is authorized to delegate administration if it has been assigned the *delegation permission*: the delegation permission allows one to create and administer sub-domains. It is important to notice that a member of the creator role can only delegate a subset of role permissions; therefore it cannot occur that one assigns a permission that he or she does not hold. The delegation permission, if assigned to the admin role r of a sub-domain, authorizes r to further delegate administration. Notice that this mechanism is similar to the mechanism of *grant option* in discretionary models, in that the subject who has the delegation permission is authorized to further propagate the administration delegation.

3.4 Objects and domain scoping rules

A domain is associated with a set of objects. Objects consist of *application objects* and *system objects*. The former are the information resources of the application;the latter are the objects which are created and maintained by domain administrators.

An issue that is generally ignored in literature concerns the rules which govern the visibility and thus accessibility of objects across domains. The problem is exemplified as follow: can an object created in a domain d, say a role named $doctor$, be accessed from domain d'? Since this problem resembles the definition of scoping rules in programming languages, we refer to these criteria of visibility as *domain scoping rules*. Note that we only consider the issue of visibility for system objects. We have identified three possible approaches. A first approach is to have a pool of objects, visible from all the application domains, thus *global*, and administered by the members of Top Admin. The drawback of this approach is the limited autonomy of the domain administrator in the management of the domain policy. A second approach is to assume that regular objects are *local* to each domain. This approach ensures the maximum autonomy; the drawback is that the objects which are of common concern for the organization and that could be shared or reused must be replicated in each domain. The third approach is a compromise of the previous two and is the one adopted in GEO-RBAC Admin: a subset of the system objects are global while the remaining system objects are local. The practical effect of this domain scoping rule is the following: when the user connects to the system and specifies the domain in which he or she is registered, the objects which become available are those local to the domain and those which are globally defined.

In principle, the distinction of what is global or local could exclusively depend on the application needs and the demand of re-usability of system objects across domains. In practice, the choice can be also driven by usability concerns. This consideration results from the experiments we have carried out with the prototype of the administration system. This prototype enables the creation of various elements of the domain policy which are then stored in a spatial DBMS. It turns out that some objects may be complex to specify for a "normal" administrator: for example the specification of a *location mapping function*, which ideally can be whatever function returning a logical position out of the real position, entail programming capabilities that administrators do not necessarily have. It seems thus more convenient to let the Top Admin administer these objects, under the assumption that the member of such a role has the needed competences. In conclusion, domain scoping should result from the desired trade-off between simplicity of use and administrative autonomy. In its present version, the domain scoping rules are cabled in the model. A more flexible management of domain scoping rule however would be desirable. This aspect will be investigated as part of future plans.

In the next section we provide a formal definition of the GEO-RBAC Admin model.

4 Specification of GEO-RBAC Admin

4.1 Objects

System object class	Meaning
C_R	Regular roles class
C_{RS}	Regular role schemas class
C_{AR}	Administrative roles class
C_{ARS}	Administrative role schemas class
C_{EXT}	Role extent types and instances
C_{LPS}	Logical position types and instances
C_{LMF}	Logical mapping functions
C_U	Regular users class
C_{AU}	Administrative users class

Table 1. System classes

Objects are grouped in a set of predefined classes. Classes which comprise system objects are *system* classes, otherwise *application* classes. System classes in turn consist of *regular* and *administrative* classes. System classes, reported in Table 1, are application independent. It can be noticed that system classes include not only the class of role schemas but also the various classes of schema components. The choice of the granularity of objects, in particular composite objects, that is objects consisting of parts like role schemas is an important design issue. The set of application classes is application dependent. Without significant loss of generality we assume a unique class denoted as C_{AP}.

4.2 Permissions

Access modes A key design choice concerns the definition of the *access modes*. At one extreme, one could define an access mode for each administrative operation. Unfortunately the resulting set could be redundant. The choice is thus to define access modes only for the operations which modify the state following the principle of *reversibility*[16]. Such a principle states that a permission should enable the reversibility of an action, for example the permission for creating on object should also authorize the deletion of the object. The set AM of access models over administrative objects is thus defined as follows:

$$AM = \{admin, assignPrm, assignUser, delegate\}$$

where: *admin* allows one to create and delete an object; *assignPrm* is to assign and revoke permissions on roles and role schemas; *assignUser* is to assign and revoke users to roles; *delegate* is to authorize and revoke administration delegation.

The set of access mode over application objects is application dependent.

Permissions. Permissions comprise *application permissions* denoted as $PRMS_A$ and *system permissions* denoted as $PRMS_S$.

Permission	Meaning
$[C_R, admin]$	- to create and delete a regular role
$[C_{RS}, admin]$	- to create and delete a regular role schema
$[C_R, assignPrm]$	- to assign and revoke regular permissions to regular roles
$[C_{RS}, assignPrm]$	- to assign and revoke regular permissions to regular role schemas
$[C_R, assignUser]$	- to assign and revoke regular users to regular roles
$[C_{EXT}, admin]$	- to create and delete role extent types and instances
$[C_{LPS}, admin]$	- to create and delete logical position types and instances
$[C_{LMF}, admin]$	- to create and delete logical mapping functions
$[C_U, admin]$	- to create and delete regular users
$[C_{AR}, delegate]$	- to create and delete a domain, an admin schema, an admin role for the created domain; to assign and revoke a user to the admin role, a permissions to an admin role

Table 2. The set $PRMS_S$ of system permissions

System permissions take the form: $[c, a]$ where c is a system class and a an access mode. For example $[C_R, admin]$ is the permission of creating and deleting regular roles. System permissions are listed in Table 2. In general, each access mode authorizes two or more operations.

In particular the *delegate* access mode authorizes the whole set of operations which are needed to delegate (or revoke) administration to the admin roles of a sub-domain. Since authorizing each of these operations separately would be useless because they are all part of a unique workflow, a unique permission is introduced for the whole set of operations, the *delegation permission* $[C_{AR}, delegate]$.

Unlike system permissions, application permissions are application-dependent. Such permissions can be defined over a class of application objects or over a single application object. In the latter case, permissions are local to the domain.

4.3 Domains

A domain d is represented by the tuple:

$$d \equiv < id, n, s >$$

where id the domain identifier, n is the domain name, and s is spatial feature of system defined type representing the reference space. Note that the domain identifier is needed since the same name can be used for domains created by

different administrators. Since domains have a hierarchical structure, the unique identifier can take the form of absolute path along the hierarchy of domains.

We denote with *TopDomain* the system-defined domain: $TopDomain \equiv < id_0, TopDomain, TopF >$, where $TopF$ is the system-defined feature which covers the whole Earth. D denotes the set of domains.

Example 1 *Suppose that Italy and Switzerland are the identifiers of two spatial features of system-defined type. Further, suppose that we would like to define two domains, one for each country, say MobileI and MobileCH. The two domains take the form:* $< id_1, MobileI, Italy >$ *and* $< id_2, MobileCH, Switzerland >$

4.3.1 State of a domain

Each domain has a *state*. The state defines the sets and relations which hold in the domain at a certain stage. A domain is described at intensional level by a domain schema and at extensional level by domain states (simply *state*). We introduce the following notation: C to denote the set of system classes; C_{AP} to denote the unique application class; O to denote the set of all possible objects. Moreover, function $OBJS(c)$ returns the set of possible objects in O of class c.

Definition 1 (Schema of domain state) *The state schema of a domain is defined by the following tuple of functions:*

- $OB_A : D \rightarrow 2^O$ *such that* $OB_A(d) \subseteq OBJS(C_{AP})$. *The function returns the set of application objects in a domain*
- $OB_S : C \times D \rightarrow 2^O$ *such that* $OB_S(c, d) \subseteq OBJS(c)$. *The function returns the set of system objects of a certain class in a domain.*
- $P_A : D \rightarrow 2^{PRMS_A}$ *returns the set of application permissions in a domain.*
- $P_S : D \rightarrow 2^{PRMS_S}$ *returns the set of system permissions in* d *in a domain.*
- $SUA : D \rightarrow 2^{OBJS(C_U) \times OBJS(C_R)}$ *such that* $SUA(d) \subseteq OB_S(C_U, d) \times OB_S(C_R, d)$. *The function returns the set of pairs: regular user-regular role in a domain.*
- $ASUA : D \rightarrow 2^{OBJS(C_U) \times OBJS(C_{AR})}$ *such that* $ASUA(d) \subseteq OB_S(C_{AU}, d) \times OB_S(C_{AR}, d)$. *The function returns the set of pairs: admin user-admin role in a domain.*
- $SPA_I : D \rightarrow 2^{OBJS(C_R) \times PRMS_A}$ *such that* $SPA_I(d) \subseteq OB_S(C_R, d) \times P_A(d)$. *The function returns the set of pairs: regular role-permission in a domain.*
- $SPA_S : D \rightarrow 2^{OBJS(C_{RS}) \times PRMS_A}$ *such that* $SPA_S(d) \subseteq OB_S(C_{RS}, d) \times P_A(d)$. *The function returns the set of pairs: regular role schema-permission in a domain.*
- $ASPA_I : D \rightarrow 2^{OBJS(C_{RS}) \times (PRMS_S \bigcup PRMS_A)}$ *such that* $ASPA_I(d) \subseteq OB_S(C_{AR}, d) \times (P_S(d) \bigcup P_A(d))$. *The function returns the set of pairs: admin role-permission in a domain.*
- $ASPA_S : D \rightarrow 2^{OBJS(C_{ARS}) \times (PRMS_S \bigcup PRMS_A)}$ *such that* $ASPA_S(d) \subseteq OB_S(C_{ARS}, d) \times (P_S(d) \bigcup P_A(d))$. *The function returns the set of pairs: admin role schema-permission in a domain.*

A *domain state instance* (simply *domain state* or *state*), denoted as $\mathcal{I}$(d) is the state schema applied to domain d. Conventionally the admin roles in a domain d have as extent the reference space of d. We denote with $\mathcal{I}$ the *state of the domain set*, defined a $\mathcal{I} = \{\mathcal{I}(\text{d}), d \in D\}$

The *TopDomain* is such that each state $\mathcal{I}$(TopDomain) has the properties described below.

Definition 2 (State of the TopDomain) *The state of the TopDomain satisfies the following properties:*

- $P_A(TopDomain) = PRMS_A$
- $P_S(TopDomain) = PRMS_S$
- $OB_S(C_{AR}, TopDomain) = \{TopAdmin\}$
- $OB_S(C_{AU}, TopDomain) \supseteq \{TopUser\}$
- $OB_S(C_o, TopDomain) = \emptyset$ *with* $o \in \{C_R, C_U, C_{RS}\}$
- $ASPA_I(TopDomain) = \{TopAdmin\} \times (PRMS_A \bigcup PRMS_S)$
- $ASUA(TopDomain) \supseteq \{[TopDomain, TopUser]\}$

4.3.2 Admin hierarchy

An admin role ar of a domain d can create sub-domains and delegate administration to admin roles in the newly created sub-domains. Given a domain d_2 created by an admin role ar of a domain d_1 we call d_1 the *parent* domain of d_2 (d_2 is sub-domain of d_1) and role ar the *owner* of d_2. By definition the owner of a domain is unique. We recall the key design choice, that the owner of a domain is the only admin role that can administer that domain and the corresponding admin roles and admin users.

The domain ownership relationship is represented by the *Admin Hierarchy*. The Admin Hierarchy consists of a partially ordered set of *nodes* where nodes take the form $[d, ar]$ with d a domain and ar an admin role in d or $\perp$ (undefined).

Definition 3 (Admin hierarchy) *Let* $\mathcal{I}$ *be the state of the domain set. The Admin Hierarchy* $AH = (T, \prec)$ *in* $\mathcal{I}$ *is such that:*

(1) $T \subseteq \{[d, ar] \mid d \in D,\ ar \in OB_S(C_{AR}, d) \bigcup \{\perp\})$

(2) $\prec \subseteq T \times T$ *is a partial order. The ordering* $[d_m, r_m] \prec [d_n, r_n]$ *holds iff* $d_m \neq d_n$, $r_m \neq \perp$ *and one of the following two conditions is true:*

- r_m *is the owner of* d_n

– $\exists [d, r] \in T$ *such that* $[d_m, r_m] \prec [d, r] \prec [d_n, r_n]$

(3) $\forall\, [d, ar] \in T,\ [TopDomain, TopAdmin] \prec [d, ar]$

It can be easily shown that the admin hierarchy constitutes a tree. In fact each node represents an admin role for a domain. Since by definition, an admin role in a domain can be only created by an admin role in another domain, it follows that every node, different from the root, has a unique parent node. Therefore since the set of ancestors of a node is totally ordered and has a bottom element, the partial order is a tree.

Example 2 *Assume that TopAdmin has created two domains. The rectangles in Figure 1 correspond to domains:* dom_1*, and* dom_2 *with reference space* s_1 *and* s_2 *respectively. Moreover the Top Admin has created two admin roles, one for each of the newly created domains, named respectively* $ar(s_1), ar(s_2)$*. Notice that these two admin roles are instances of the same schema* ar*. The member of the former role has created two sub-domains,* dom_{11}, dom_{12} *over space* s_{11} *and* s_{12} *respectively. The former is administered by two admin roles:* $ar_1(s_{11}), ar_2(s_{11})$*; the latter by a unique role* $ar_1(s_{12})$*. The is illustrated in Figure 1.*

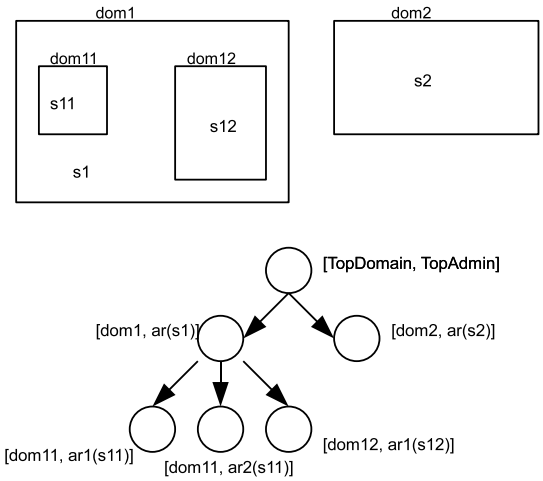

Figure 1. Admin hierarchy

Finally, we introduce the notion of *global state*, defined by the pair $\mathcal{I}^* =< \mathcal{I}\ , AH >$.

4.3.3 Constraints on states

Administrative operations determine a change of state. Operations can lead, however, to an improper state. For example, if a role schema is removed from a domain while instances of that schema still exist, then the enforcement of those roles would be impossible or fail in any case. In that case we say that the state of the domain is *unsound*. It may also occur that some necessary administration operation has not been performed yet and thus a domain is in a transitory state. In that case, we say that the state of the domain is *administratively incomplete*. For example, if in a domain there is a unique admin role which is authorized to administrate regular roles and that admin role is removed, then the regular roles which still exist cannot be administered any longer until a new administrator is appointed.

The understanding of the conditions which must hold for domains to be sound and administratively complete is a necessary step for the definition of administration operations. For that purpose, we explicitly represent these conditions in terms of constraints. In what follows we introduce diverse sets of constraints.

4.3.4 Soundness of domains and admin hierarchy

Let $\mathcal{I}$ be the state of the domain set.

Definition 4 (Soundness of domain) *A domain* $d \in D$ *is sound in* $\mathcal{I}$ *iff it holds:*

a) Regular and admin roles in d *have an extent contained in the reference space of* d*.*

b) Each regular role is instance of a regular role schema in d*, that is,* $\forall r \in OB_S(C_R, d), \exists rs \in OB_S(C_{RS}, d),\ rs = Sk(r)$*.*

The first condition implies that roles cannot be played outside the domain boundaries since domains constrain the spatial extent in which the movement of users and their operations is significant for the domain policy. The second condition is to prevent role instances from being *orphans* of their schema. We now define when the Admin Hierarchy is sound. Preliminarily we introduce the following functions: $d\_Prms(r, d)$ returns the set of permissions assigned to admin role r in domain d comprehensive of the permissions assigned to the role schema or directly to r; $Owner(d)$ returns the admin role which is owner of d; $Parent(d)$ returns the domain of the owner role.

Definition 5 (Soundness of Admin Hierarchy) *Admin Hierarchy* $AH = (T, \prec)$ *is sound in* $\mathcal{I}$ *iff for each domain* $d \in D - \{TopDomain\}$ *it holds:*

c) It exists at least one node $[d, r] \in T$ *for some value of* r*.*

d) The owner role of d *has the permission* $[C_{AR}, delegate]$*, that is:* $(Owner(d), [C_{AR}, delegate]) \in ASUA(Parent(d))$*.*

e) The reference space of d *is spatially contained in the reference space of the parent domain.*

f) Permissions assigned to an admin role ar in d are included in the set of permissions assigned to the owner of d, that is: $\forall p \in d\_Perm(ar, d) \rightarrow p \in d\_Perm(Owner(d), Parent(d))$.

Condition c) means that there are no domains which are isolated. Condition d) means that there is at least one admin role in the parent domain which is authorized to delegate administration. Condition e) implies that a domain is spatially confined by the space of the parent domain. The last condition is to prevent one from delegating permissions that he or she does not have. Note that the delegated permissions can be only defined on classes of objects and not on single objects, since these are local to each domain. It is easy to show that a sound Admin Hierarchy has the following properties:

Proposition 1 (Properties of admin hierarchy) *Let $AH = (T, \prec)$ be a sound admin hierarchy in $\mathcal{I}$.*

- *Given two domains d_1 and d_2, it holds:*
  - *If d_1 and d_2 are not comparable, in that one domain is not the ancestor of the other then the respective reference spaces, $d_1.s$ and $d_2.s$ can be spatially disjoint or overlap.*
  - *The set of admin roles in d_1 and d_2 as well as the corresponding sets of admin users are disjoint, that is: $OB_S(C_{AR}, d_1) \bigcap OB_S(C_{AR}, d_2) = \emptyset$*
- *The value of the admin role in node $[d, ar] \in T$ may be undefined only if the node is a leaf in AH.*

4.3.5 Administration completeness

Some objects are global, because of domain scoping rules and thus are administered by the top administrator. Domain scoping rules are expressed as constraints as follows:

Definition 6 (Domain scoping rules) *Forall $d \in D - \{TopDomain\}$ it holds:*

- $OB_S(C_{EXT}, d) = OB_S(C_{EXT}, TopDomain)$
- $OB_S(C_{LPS}, d) = OB_S(C_{LPS}, TopDomain)$
- $OB_S(C_{LMF}, d) = OB_S(C_{LMF}, TopDomain)$

The state of a domain which is sound is also administratively complete if such domain has at least one member of an admin role and every object which is local to the domain is administered. The following conditions formally define the administration completeness property.

Definition 7 (Administration completeness) *Let $d \in D$ a domain which is sound in $\mathcal{I}$. Then d is also administratively complete iff it holds:*

- *Each admin role has been assigned at least one admin user.*
- *The owner role of a domain has at least one admin user.*
- *For each regular role and regular user o, there must be at least one admin role $ar \in OB_S(C_{AR}, d)$ which is authorized to administer o, that is $(ar, [C_o, admin]) \in ASPA_I(d)$.*
- *For each applicative object $o \in OB_A(C_{AP}, d)$, there must be an admin role that administers o.*

Finally, consider a global state. It can be easily shown that the following property holds:

Proposition 2 *Let $\mathcal{I}^* = < \mathcal{I}, AH >$ be a global state. If each domain $d \in D$ is administratively complete, and AH is sound, then for every object o, it exists an admin user who is authorized to administer o.*

4.4 Administrative operations

Table 3 reports the administrative operations. In this section, for the sake of space, we only describe the process of delegating administration to a sub-domain using the operations of *CreateDomain, CreateAdmObj, GrantAdmPrm, GrantAdmUser*.

Administration delegation: workflow Assume that the member of an admin role ar in current domain cd wants to create a sub-domain sd. 1) As first step ar creates a sub-domain through the operation *CreateDomain*. A child of node $[ar, cd]$ which is only partially filled is created and added to the admin hierarchy. 2) If not already existing, ar creates an admin schema ars' in the current domain through the operation *CreateObj* and possibly assigns permissions to that schema through the operation *GrantAdmPrm*. 3) ar creates an admin role ar' as instance of schema ars' through the operation *CreateAdmObj* and possibly assigns permissions through *GrantAdmPrm*: ar' is thus automatically added to the set of admin roles of sub-domain sd. The admin hierarchy is updated.4) Finally ar creates an admin user au' through *CreateAdmObj* and assigns it to the admin role ar' through *GrantAdmUser*.

Notice that if the delegation permission is then assigned to the admin role of the newly created sub-domain, then the administration can be further delegated to nested sub-domains. Further, observe that the creator of a sub-domain can also add new admin roles after the domain is created.

5 Conclusions

In this paper we have presented a model for the decentralized administration of GEO-RBAC based on hierarchy of spatial domains. The approach is quite flexible. Moreover,

Admin object class	meaning
$CreateRegObj(c, o)$	Create a regular object o of class $c \in \{C_R, C_{RS}, C_U\}$
$DeleteRegObj(c, o, d)$	Delete a regular object o of class $c \in \{C_R, C_{RS}, C_U\}$
$CreateAdmObj(c, o, d)$	Create an admin object o of class $c \in \{C_{AR}, C_{ARS}, C_{AU}\}$ in domain d
$DeleteAdmObj(c, o, d)$	Delete a admin object o of class $c \in \{C_{AR}, C_{ARS}, C_{AU}\}$ in domain d
$CreateDomain(d, s)$	Create a domain d with reference space s as sub-domain
$DeleteDomain(d)$	Delete domain d
$CreateSchemaElement(c, e)$	Create a role schema element e of class $c \in \{C_{EXT}, C_{LPS}, C_{LMGF}\}$:
$DeleteSchemaElement(c, e)$	Delete a schema element e of class $c \in \{C_{EXT}, C_{LPS}, C_{LMGF}\}$
$GrantRegPrm(c, r, p)$	Assign application permission p to regular object r of class $c \in \{C_R, C_{RS}\}$
$RevokeRegPrm(c, r, p)$	Revoke application permission p to regular object r of class $c \in \{C_R, C_{RS}\}$
$GrantAdmPrm(c, r, p, d)$	Assign permission p to admin object r of class $c \in \{C_{AR}, C_{ARS}\}$ in domain d
$RevokeAdmPrm(c, r, p, d)$	Revoke permission p to admin object r of class $c \in \{C_{AR}, C_{ARS}\}$ in domain d
$GrantRegUser(u, r)$	Assign regular user u to regular role r
$RevokeRegUser(u, r)$	Revoke regular user u to regular role r
$GrantAdmUser(u, r, d)$	Grant admin user u to admin role r in domain d
$RevokeAdmUser(u, r, d)$	Revoke admin user u from admin role r in domain d
$Display(d, \{c_1, ..c_n\})$	Display on map the confine of domain d along with the spatial features of type $c_1, .., c_n$

Table 3. Administration operations

although nested domains can be freely created, the administration still remains indirectly under the control of the Chief Security Officer and that is important in contexts demanding strong access control. Some important issues are still open and will be investigated as part of the future activity:

- **User sharing.** In GEO-RBAC Admin, a users u who is registered in domain d is not visible in a sub-domain d'. The question is thus how to enable a more flexible management of users, so to allow a sort of automatic registration in sub-domains. To that purpose, a possible approach is to *share users* between domains.

- **Administrative object sharing**. An extension of the previous functionality is to enable the sharing of additional administrative objects, in particular role schemas. That would allow the specification of a role ontology for the organization.

- **Mobility**. The issue is how to support mobility of users across domains, enabling a transparent connection to heterogeneous domains.

References

[1] C. Ardagna, M. Cremonini, E. Damiani, S. D. C. di Vimercati, and P. Samarati. Supporting Location-based Conditions in Access Control Policies. In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communication Security*, 2006.

[2] E. Bertino, P. A. Bonatti, and E. Ferrari. TRBAC: a temporal role-based access control model. *ACM Transaction on Information Systems Security*, 4(3):191–233, 2001.

[3] E. Bertino, B. Catania, M. Damiani, and P. Perlasca. GEO-RBAC: a spatially aware RBAC. In *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies (SACMAT'05)*, pages 29–37, Stockholm, Sweden, 2005. ACM Press.

[4] E. Bertino and R. Sandhu. Database Security-Concepts, Approaches, and Challenges. *IEEE Transactions on Dependable and Secure Computing*, 2:2–19, 2005.

[5] R. Bhatti, J. B. D. Joshi, E. Bertino, and A. Ghafoor. X-GTRBAC Admin: A Decentralized AdministrationModel for Enterprise-Wide Access Control. *ACM Transactions on Information and System Security*, 4, 2005.

[6] M. Covington, W. Long, S. Srinivasan, A. Dev, M. Ahamad, and G. Abowd. Securing context-aware applications using environment roles. In *Proceedings of the 6th ACM symposium on Access control models and technologies (SACMAT'01)*, pages 10–20, Chantilly, Virginia, USA, 2001. ACM Press.

[7] J. Crampton and G. Loizou. Administrative scope: A foundation for role-based administrative models. *ACM Trans. Inf. Syst. Secur.*, 6(2):201–231, 2003.

[8] M. Damiani, E. Bertino, B. Catania, and P. Perlasca. GEO-RBAC: A spatially Aware RBAC. *ACM Transactions on Information and System Security (TISSEC)*, 10(1), 2007.

[9] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, August 2001.

[10] M. Ge and S. L. Osborn. A Design for Parameterized Roles. In F. C. and S. P., editors, *Research Directions in Data and Applications Security XVIII, IFIP TC11/WG 11.3 Eighteenth Annual Conference on Data and Applications Security*, 2004.

[11] L. Giuri and P. Iglio. Role Templates for Content-Based Access Control. In *ACM Workshop on Role Based Access Control*, 1997.

[12] P. P. Griffiths and B. W. Wade. An authorization mechanism for a relational database system. *ACM Trans. Database Syst.*, 1(3):242–255, 1976.

[13] R. Hulsebosch, A. Salden, M. Bargh, P. Ebben, and J.Reitsma. Context-Sensitive Access Control. In *In* Proceedings of the tenth ACM symposium on Access control models and technologies*, June, 2005, Sweden.*, 2005.

[14] A. Kern, A. Schaad, and J. Moffet. An Adminstration Concept for the Enterprise Role-based Access Control Model. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies*, 2003.

[15] M. Kumar and R. Newman. STRBAC - An approach towards spatio-temporal role-based access control. In *Communication, Network, and Information Security*, pages 150–155, 2006.

[16] N. Li and Z. Mao. Administration in role-based access control. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 127–138, New York, NY, USA, 2007. ACM Press.

[17] J. Moffet. *Delegation of Authority Using Domain-Based Access Rules*. PhD thesis, Imperial College of Science, Technology & Medicine, 1990.

[18] S. Oh, R. Sandhu, and X. Zhang. An effective role administration model using organization structure. *ACM Trans. Inf. Syst. Secur.*, 9(2):113–137, 2006.

[19] R. Sandhu, V. Bhamidipati, and Q. Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transaction on Information Systems Security.*, 2(1):105–135, 1999.

[20] A. Schaad, J. Moffett, and J. Jacob. The role-based access control system of a European bank: a case study and discussion. In *Proceedings of the 6th ACM symposium on Access control models and technologies*, pages 3–9, New York, NY, USA, 2001. ACM Press.