CERIAS Tech Report 2003-24

OPTIMAL SECURE INTEROPERATION IN A MULTI-DOMAIN ENVIRONMENT EMPLOYING RBAC POLICIES

by Basit Shafiq, James B. D. Joshi,
Elisa Bertino, Arif Ghafoor

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907

Optimal Secure Interoperation in a Multi-Domain Environment Employing RBAC Policies

Basit Shafiq[1], James B. D. Joshi[1], Elisa Bertino[2], and Arif Ghafoor[1]
[1]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN
{shafiq, joshij, ghafoor}@ecn.purdue.edu
[2]Dipartimento di Scienze dell'Informazione, Università di Milano, Milano, Italy

Abstract

Multi-domain application environments where distributed multiple organizations interoperate with each other are becoming a reality as can be seen in most Internet-based enterprise applications. Composition of a global security policy that governs information and resource accesses in such environments is a challenging problem. In this paper, we propose a policy integration mechanism that merges security policies of multiple collaborating domains into one unified global access control policy. This global policy ensures that security and autonomy of constituent domains are not compromised due to inter-domain information and resource sharing.

*Portions of this work were supported by Grant IIS-0209111 from the National Science Foundation, and by sponsors of the Center for Education and Research in Information Assurance and Security.

1 Introduction

Recent advances in high-performance computing and networking technologies have resulted in a tremendous growth of large-scale distributed applications in medicine, education, e-commerce, digital libraries, digital government, ubiquitous computing environments and many others. With such rapid proliferation and increasing importance of IT technologies, security is becoming a major concern. Many studies show that unauthorized access, in particular by *insiders*, constitutes a major security problem for enterprise application environments [Pow00], highlighting the need for robust access control management systems. This problem can be highly magnified in a multi-domain environment where distributed multiple organizations, each employing its own security policy, interoperate with each other, allowing highly intensive inter-domain accesses [Jos01b, Gon96]. Such multi-domain environments are already becoming a reality, as can be seen in most Internet-based enterprise applications, digital governments, and healthcare systems [Jos01b]. Each individual domain of a multi-domain environment can have its own security policy. Integration of these local policies entails various challenges regarding reconciliation of semantic differences between local policies, secure interoperability, containment of risk propagation, policy management, etc. [Jos01b]. An access control model that can be used to uniformly represent policies of the individual domains is desirable. Such a model should allow interoperation and information sharing among multiple domains and at the same time guarantee that such inter-domain accesses do not violate the underlying policies of the constituent domains.

A multi-domain system can be considered as a collection of cooperating single domain systems which are possibly autonomous and heterogeneous. Heterogeneity in a multi-domain environment may exist in different forms [Hos91]. For instance, the multi-domain environment may be composed of a diverse set of interacting and collaborating organizations with different policies. Similarly, the environment may have more than one security goals, or the variations of the same goal. Furthermore, the environment may have heterogeneous system components such as operating systems and databases, each with different security policy [Hos91, Jos01b].

In a multi-domain system, information and resources belonging to a particular domain can be accessed by outside users. However, these inter-domain accesses should not result in any violation of the security policies of constituent domains. In particular, secure interoperation should enforce the following two principles [Gon96]:

- The *autonomy principle*, which states that if access is permitted within an individual system, it must also be permitted under secure interoperation.
- The *security principle*, which states that if an access is not permitted within an individual system, it must not be permitted under secure interoperation.

To achieve these security goals, a unified policy called *meta-policy* is needed to mediate accesses across domain boundaries [Hos91]. Such meta-policy is responsible for resolving semantic differences and inconsistencies among the security policies of different domains. Moreover, the access control mechanism enforcing such meta-policy should also allow a domain to leave the multi-domain environment without creating access control loopholes in both itself and the overall environment it belonged to.

This paper addresses the issue of policy integration in a multi-domain system that allows secure interoperation. The security policies of collaborating domains are expressed using role-based access control (RBAC) model. The policy integration mechanism described in this paper is a two phase process as shown in figure 1. In the first phase the role heterogeneity constraints among collaborating domains are resolved and a global access control policy is generated from the given RBAC policies and administrator specified constraints. The global policy generated in the first phase may be conflicting and may allow violation of some of the security requirements. In the second phase, conflicts are resolved by relaxing some of the access constraints. For conflict resolution, we propose an integer programming (IP) [Wol98] based approach that maximizes inter-domain information and data exchange according to some specified optimality criterion.

The paper is organized as follows. In section 2, we describe related work. In section 3, we briefly discuss RBAC model. Policy integration mechanism is presented in section 4. Section 5 concludes the paper and provides future directions.

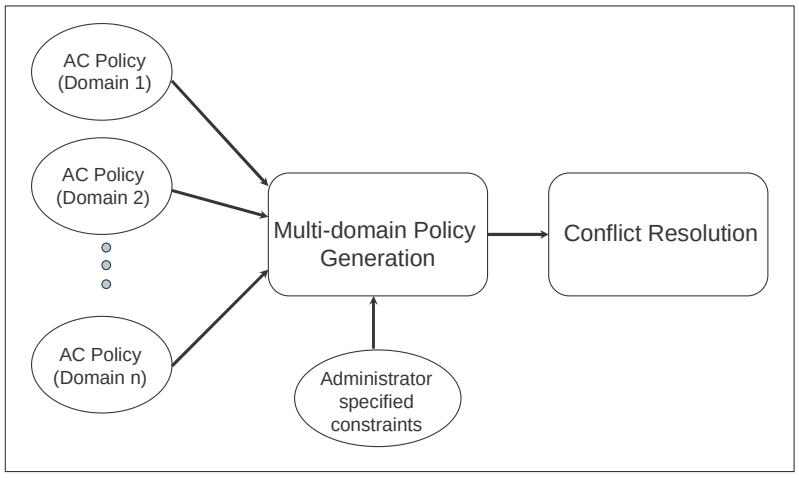

Figure 1. Policy integration framework

2 Related Work

It has been proven that general security problem in a single domain is undecidable [Har76]. Gong *et. al* [Gon96], further proved that general secure interoperation problem in a multi-domain environment is also undecidable. However, a restricted version of secure interoperation problem is shown to be tractable. In particular, secure interoperation has been investigated in the context of multi-level security (Bell Lapadula) model and is proved to be decidable in polynomial time [Gon96, Bon96]. Multi-level security or

Bell Lapadula [Bel73] model is more suitable for environments which have static constraints. For instance, in multi-level security model, all accesses conform to the pre-specified security ordering. Security ordering in this case is a static constraint, even though the security labels of entities may change with time, e.g., declassification of documents after a certain period of time. If a subject *s* with security level *a* is authorized to access an object *o* with security level *b*, then *s* can access *o* at all times provided the security levels of *s* and *o* never change. Dynamic constraints on the other hand, may not allow subject *s* to access *o* even though their security labels remain unchanged. Separation-of-duty (SoD) and precedence constraints are example of such dynamic constraints and are required in most commercial applications including e-commerce, health-care systems, and workflow management systems [Ber99]. Such applications cannot be modeled using traditional multi-level (LBAC) model because of the presence of dynamic constraints. Role based access control (RBAC) that allows specification of a wide variety of constraints including SoD, cardinality, and dependency, can be used to model these applications [Ber99]. Osborn in [Osb02] discusses integration of RBAC policies in a multi-domain environment. Policy integration in [Osb02] is performed based on user to role and role to permission assignments only. It does not take into account constraint heterogeneity that may exist among inter-domain roles.

3 Role based Access Control

Role based access control (RBAC) has been found to be a suitable model for the specification of security requirements of commercial applications [Ber99]. Several beneficial features such as policy neutrality, support for least privilege and separation of duty/privilege are associated with RBAC models [Jos01a, San96]. Role hierarchies in RBAC also allow modeling of multi-level security policy (Bell Lapadula [Bel73], Biba [Bib77]). Furthermore, RBAC constraints allow specification of user-specific access control policies, as well as discreationary access control (DAC) and mandatory access control (MAC) policies, thus, increasing the applicability of RBAC models [Osb00].

The RBAC model as proposed by Sandhu *et. al.* in [San96], currently being used as the basis for the NIST RBAC model, consists of the following four basic components: a set of *users*, a set of *roles*, a set of *permissions*, and a set of *sessions*. A user is a human being or a process within a system. A role is a collection of permissions associated with a certain job function within an organization. Permission defines the access rights that can be exercised on a particular object in the system. A session relates a user to possibly many roles. When a user logs in the system he establishes a session by activating a set of enabled role that he is entitled to activate at that time. If the activation request is satisfied, the user issuing the request obtains all the permissions associated with the role he has requested to activate. On *roles*, a hierarchy is defined by $\geq$. If $r_i \geq r_j$, and $r_i, r_j \in$ *roles* then r_i inherits the permissions of r_j. In such a case, r_i is a senior role and r_j a junior role.

3.1 Graph-based specification model for RBAC

We use graph based formalism similar to one described in [Koc02] to specify the RBAC policy of a domain. In this graph based model, users, roles, and permissions are represented as nodes and the edges of the graph describe the association between various nodes. In order to capture the RBAC semantics, the nodes can not be connected in an arbitrary manner. The type graph shown in figure 2**,** defines all possible edges that may exist between different nodes. An edge between user node *u* and role node *r* indicates that role *r* is assigned to user *u*. Self edges on the role node *r* models the role hierarchy. An edge from a role r_1 to a role r_2 indicates that role r_1 is senior to role r_2 and can inherit all permissions associated with role r_2. Furthermore, there can be edges between role and permission nodes. A permission is a pair (*access mode*, *object*), which describes what objects can be accessed and in which mode (read, write, execute etc).

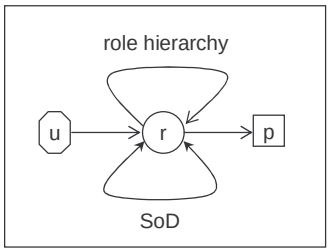

Figure 2. Type graph for the RBAC model

The graph model also supports specification of separation of duty (SoD) constraints among roles. A role specific SoD constraint disallows assignment of conflicting roles to same user. In the graph model, SoD constraint between two roles is represented by a double arrow between the corresponding roles. Figure 3 shows an instance of an RBAC policy specified in graph-based formalism. In this RBAC policy, roles r_2 and r_3 are conflicting roles and cannot be accessed by same user.

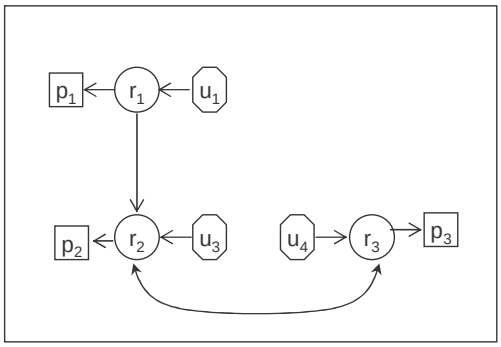

Figure 3. An RBAC policy instance

4 Multi-domain policy integration

In the following, we first discuss the heterogeneity issues involved in multi-domain policy integration and then present the proposed policy integration mechanism.

4.1 Heterogeneity issues in policy integration

Composition of a global policy that governs interoperation in a multi-domain system is a challenging problem. One key aspect of this complex problem is heterogeneity. There are various types of heterogeneity that need to be addressed as a part of policy integration procedure. The heterogeneity may arise because of naming conflicts, role hierarchies or other constraints. Naming conflicts arises because of the use of same names to represent different conceptual entities (homonym) or different names to represent same conceptual entities (synonym). Accordingly, there may be naming conflicts among inter-domain roles or objects. Naming conflicts can be resolved using schema integration techniques from the database area [Gua02, Vet98]. These techniques require the use of a global lexicon to extract the conceptual meaning of attributes from their names. Additionally, domain-based and value-set-based comparisons can be performed for refinement [Li94]. Since a role is a collection of objects/permissions, resolving naming conflicts at the role level is difficult. Therefore, naming conflicts should be resolved at the object/permission level. Once the naming heterogeneity is resolved at the object/permission level, roles from different domains can be compared and linked. Linking of roles allows inter-domain interoperation as explained in the next section.

In addition to naming conflicts, heterogeneity among multiple domains may exist in role hierarchies and in other dynamic constraints such as SoD and cardinality constraints. Hierarchical heterogeneity among domains' policies may arise because of two reasons: a) use of different role hierarchies (inheritance *I*, activation A, inheritance-activation *IA*, hybrid [Jos02]) by different collaborating domains; b) domains may use different hierarchical ordering to represent same permission authorization for a given role. For simplicity in discussion, we use a monotype inheritance only hierarchy in this paper. However, the latter hierarchical heterogeneity problem may still exist in monotype hierarchies as explained in a later section.

The objective of policy integration is to allow information and resource sharing without violating the security and autonomy of individual domains or of the multi-domain system as a whole. Since domains' RBAC policies define both permitted and restricted accesses, the security and autonomy requirements of the individual domains can be extracted from their respective RBAC policies. Additional security requirements of the multi-domain system can be defined by an administrator with global security responsibility. The administrator in charge of global security policy may specify both *permitted* and *restricted* inter-domain accesses. Note that the accesses specified by the administrator may conflict with the domain policies and may violate the security or autonomy requirements of constituent domains as well as of the merged organization. To resolve such conflicts, we propose an integer programming [Wol98] based approach that determines an optimal set of permitted accesses which preserve the specified constraints. This approach is discussed in section 4.3

4.2 Inter-domain role comparison and linking

In this section, we focus on the issue of comparing and linking inter-domain roles based on their permission set and the security and autonomy requirements of collaborating domains. As stated in the above section, inter-domain roles are compared based on their permission assignments over objects. This permission set includes both directly assigned permissions as well as inherited permissions. We also assume that objects in the RBAC model are organized into conceptual classes, e.g., account tables, insurance claims, and audit report etc. Two cross domain objects belonging to same conceptual class are considered to be semantically equivalent.

Using the above assumption and the permission assignments of roles over the objects, two roles r_A and r_B from two distinct domains A and B respectively, can have one of the following relationships:

1. *Equivalent*: r_A is equivalent to r_B ($r_A \approx r_B$), if the permission sets *Pset(r_A)* and *Pset(r_B)* of roles r_A and r_B are equivalent. The permission set *Pset(r)* of a role *r* is the set of all permissions assigned directly or indirectly to role *r*.

$$\forall i, j : class(O_{A_i}) = class(O_{B_j}) \wedge [(O_{A_i}, a) \in Pset(r_A) \Leftrightarrow (O_{B_j}, a) \in Pset(r_B)]$$

2. *Contain*: r_A contains r_B ($r_A \geq r_B$) if the permission set *Pset(r_B)* of role r_B is included in the permission set *Pset(r_A)* of role r_A. Formally:

$$\forall j \exists i : (O_{B_j}, a) \in Pset(r_B) \Rightarrow \left[(O_{A_i}, a) \in Pset(r_A) \wedge \left(class(O_{A_i}) = class(O_{B_j}) \right) \right]$$

3. *Overlap*: r_A overlaps r_B (r_A *O* r_B) if *Pset(r_A)* and *Pset(r_B)* have some common permissions. Formally:

$$\exists i, j : class(O_{A_i}) = class(O_{B_j}) \wedge [(O_{A_i}, a) \in Pset(r_A) \wedge (O_{B_j}, a) \in Pset(r_B)]$$

4. *Not related*: r_A is not related to r_B ($r_A \neq r_B$) roles r_A and r_B do not share any common permissions. Formally:

$$\neg \exists i, j : class(O_{A_i}) = class(O_{B_j}) \wedge [(O_{A_i}, a) \in Pset(r_A) \wedge (O_{B_j}, a) \in Pset(r_B)]$$

Figure 4 presents an algorithm, called *role-integrate*, that integrates inter-domain roles based on their permission assignment. *role-integrate* is a recursive algorithm that uses bottom-up strategy to establish role equivalence. The algorithm basically checks all inter-domain roles for one of the above four relations. If the roles do not share any permission, then it returns without doing anything. If the inter-domain roles say, r_1 and r_2, are equivalent then they are linked together. Linking two inter-domain roles r_1 and r_2 implies that a user say u_i, authorized for role r_1 can also access role r_2. Similarly, a user u_j authorized for role r_2 can access role r_1. An inter-domain link in the graph model is represented by a dashed double-headed arrow between two roles. A link between two inter-domain roles is established only if they are equivalent in their permission assignment (both direct and indirect) and linking them does not violate any RBAC consistency properties [Gav98]. For instance in figure 6, roles r_{4j} and r_9 are equivalent in terms of the permission assignment and also their junior roles are equivalent. Despite this permission and hierarchical equivalence, roles r_{4j} and r_9 cannot be linked because it will make role r_{4j} conflict with r_{3j} and

r_{5j}. Since all of these roles (r_{4j}, r_{3j}, and r_{5j}) belong to the same role hierarchy, they cannot have a separation of duty constraint with each other [Gav98]. The linked roles may still possess constraint heterogeneity and are not equivalent in that respect. For instance the two cross-domain roles, r_1 and r_2, may be equivalent in their permission assignment, however *they* may have different cardinality constraints. One possible solution to this problem is to take most restrictive constraints from the two roles and add them to both roles. For instance, if r_1 has a cardinality constraint of one and r_2 has a cardinality constraint of three, then the most restrictive cardinality constraint is one and should be added to both r_1 and r_2. However, in some cases adding the most restrictive constraint may result in violation of autonomy with respect to the individual domain. Nevertheless, the security condition is always preserved. On the other hand, if linked roles retain their original constraints then autonomy is preserved but security violation may occur. So, there is a trade-off between security and autonomy requirement.

The two roles r_1 and r_2 may also have a subset-superset or overlapping relationship. If r_1, contains r_2, then a junior role r_{1j} is created by calling *split* function. In the *split* function, all the permissions and junior roles common to both r_1 and r_2 are removed from r_1 and are assigned to $r_{1j.}$ After permission reassignment r_{1j} and r_2 are linked together provided linking them does not violate RBAC consistency semantics. If r_1 and r_2 overlap but none of the roles contain each other, then two new roles r_{1j} and r_{2j} are created and made junior to r_1 and r_2 respectively. Permissions and junior roles common to both r_1 and r_2 are removed from the senior roles r_1 and r_2 and assigned to the roles r_{1j} and r_{2j}. After this permission and role assignments, r_{1j} and r_{2j} are linked if possible.

Figure 5 shows the graphical representation of RBAC policies of domain A and domain B. The double headed arrow in figure 5(b) between roles r_9 and r_{10} depicts that r_9 and r_{10} are conflicting roles and cannot be assigned to same user. Figure 6 shows the integrated RBAC policy that allows interoperation between domains A and B. Note that in the integrated policy the user to role assignments do not change from the original domain policies. Moreover, the security and autonomy requirement of domains A and B are not violated. As a result of this integration, constraints are also added between cross-domain roles. For instance, SoD constraints between r_{3j} and r_9, and r_{5j} and r_9 are added. These inter-domain SoD constraints are shown as solid double-headed arrow between the corresponding roles.

Complexity of role-integrate

The algorithm role-integrate runs in polynomial time and has a complexity of $O(|P|^2)$. This is evident from the following lemma and theorem.

Lemma 1: If role graphs representing domains' RBAC policies are acyclic then the algorithm *role-integrate* terminates.

Proof: Given two acyclic role graphs to be integrated, suppose that the algorithm does not terminate, i.e., *role-integrate* is called recursively for an infinite number of times. This implies that there is a cycle in one or both of role graphs. Creation of new roles does not create any cycle as a newly created role is never

made a parent of an existing role. Therefore, the cycle must be present in the input role graph(s) which is a contradiction of our initial assumption. Hence the algorithm *role-integrate* terminates.

Theorem 1: The worst case complexity of *role-integrate* is $O(|P|^2)$, where |P| is the cardinality of the permission set.

Proof: According to the above lemma, the recursive algorithm *role-integrate* terminates. Therefore, we can build a recursive tree with each node corresponding to two roles to be compared. The predicate *not-compared-previously* in lines 4 and 7 ensures that inter-domain roles are compared only once. If |R1| and |R2| denotes the total number of roles in their respective domains, then the total number of role comparisons made by *role- integrate* while merging the two domains are |R1|x|R2|. Note that |R1| and |R2| also include newly created roles. However, no more that |P| number of roles can be created. Therefore, the complexity of role-integrate is $O(|P|^2)$.

4.3 Administrator specified constraints

The role integration algorithm described above takes two RBAC policies as input and creates an integrated policy which allows inter-domain role accesses and is homogeneous in terms of role hierarchies and permission assignment. The administrator who is in charge of the global security policy can define additional security constraints and specify both permitted and restricted inter-domain role accesses. These additional constraints may conflict with the access control policies of individual domains. For instance, in figure 6 making role r_2 senior to the role r_{7j} (shown as dashed arrow from r_2 to r_{7j} in figure 7) will violate the role specific SoD constraints between roles r_9 and r_{10} and also between r_9 and r_{10j}. Making r_2 senior to r_{7j} will enable user u_1 to access role r_9 through the role r_{7j}. Also the presence of link between role r_{3j} and r_{10} allows user u_1 to access role r_{10}. This is a violation of SoD constraint defined between roles r_9 and r_{10} in the original domain policy.

The solution to this problem is to remove either the unidirectional link ($r_2 - r_{7j}$) or links ($r_{3j} - r_{10}$) and ($r_{5j} - r_{10j}$). This raises an important question: which accesses from the set of conflicting accesses should be removed such that the autonomy and security requirements of constituent domains are not violated? Although, removing link(s) resolves conflicts in the given policy, it also changes the set of allowable accesses. We can formulate the problem of conflict resolution in a given multi-domain RBAC policy as an optimization problem with the objective of maximizing permitted accesses according to some pre-specified optimality criterion. Various optimality measures such as maximizing direct or indirect accesses or minimizing the set of relaxed constraints can be used.

In the following, we describe an integer programming based approach that determines an optimal set of allowable accesses which do not violate the specified constraints. The formulation is generic in a sense that it can work for any of the above mentioned optimality criteria. Changing the optimality measure in our formulation only requires changing the weights of the objective function.

```
Role-integrate(r1, r2)
1.  if Pset(r1) ∩ Pset(r2) = ϕ
2.      return
3.  for each rc ∈ children(r1)
4.      do if  ((Pset(rc) ∩ Pset(r2) ≠ ϕ) and not-compared-previously(rc,r2))
5.             then Role-integrate(rc,r2)
6.  for each rc ∈ children(r2)
7.      do if ((Pset(r1) ∩ Pset(rc) ≠ ϕ) and not-compared-previously(r1,rc))
8.             then Role-integrate(r1,rc)
9.  ► return without doing anything if r1 and r2 are already linked
10. if already_linked(r1,r2)
11.     then return
12. ►ri is contained in rj if for each p assigned to ri, pis also assigned to rj, and if ri has a junior role rk
    then there exist a path from rj to role rk.
13. if contained(r2, r1) and contained(r1, r2)
14.     then if linking r1 and r2 do not violate RBAC consistency properties
15.            then link(r1, r2)
16.              return
17. else if contained(r2, r1)
18.     then r1j=split(r1, common_permissions(r1,r2), common_juniors(r1,r2) )
19.          if linking r1j and r2 do not violate RBAC consistency properties
20.            then link(r1j, r2)
21.              return
22. else if contained(r1, r2)
23.     then r2j=split(r2, common_permissions(r1,r2), common_juniors(r1,r2) )
24.          if linking r1 and r2j do not violate RBAC consistency properties
25.            then link(r2j,r1)
26.              return
27. ► r1 and r2 overlap if they have at least one common directly assigned permission or at least one
    common junior
28. else if overlap(r1,r2)
29.     then r1j=split(r1, common_permissions(r1,r2), common_juniors(r1,r2) )
30.          r2j=split(r2, common_permissions(r1,r2), common_juniors(r1,r2) )
31.          if linking r1j and r2j do not violate RBAC consistency properties
32.          then link(r1j, r2j)
33.              return
34. return

split(r, com_perm, com_juniors)
1.  rj ← createrole()
2.  insert(r->childrenlist,rj)
3.  for each p ∈ com_perm
4.      do remove(r->plist, p)
5.         insert(rj->plist,p)
6.  for each rc ∈ com_juniors
7.      do remove(r->childrenlist, rc)
8.         insert(rj->childrenlist, r)
9.         return rj
```

Figure 4. *role-integrate*: an algorithm for integrating inter-domain roles

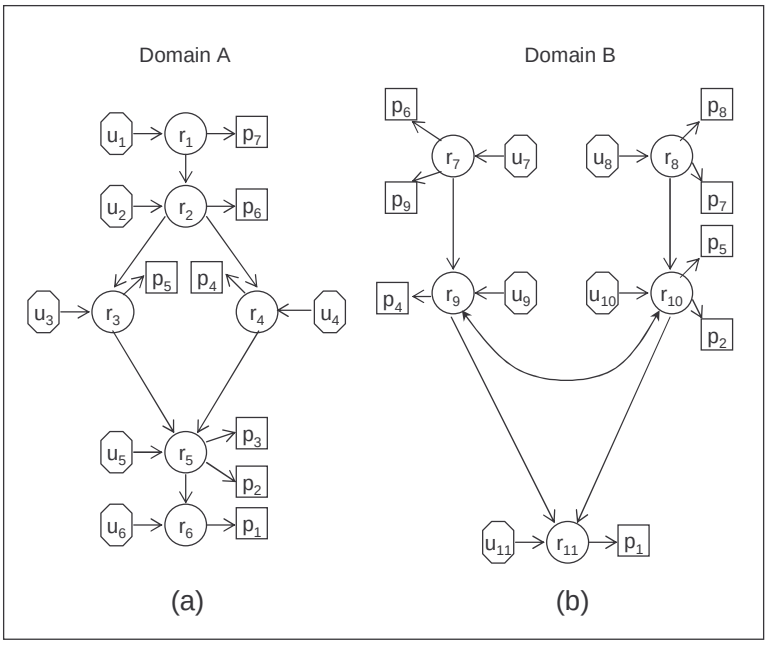

Figure 5. RBAC Policies of Domains A and B

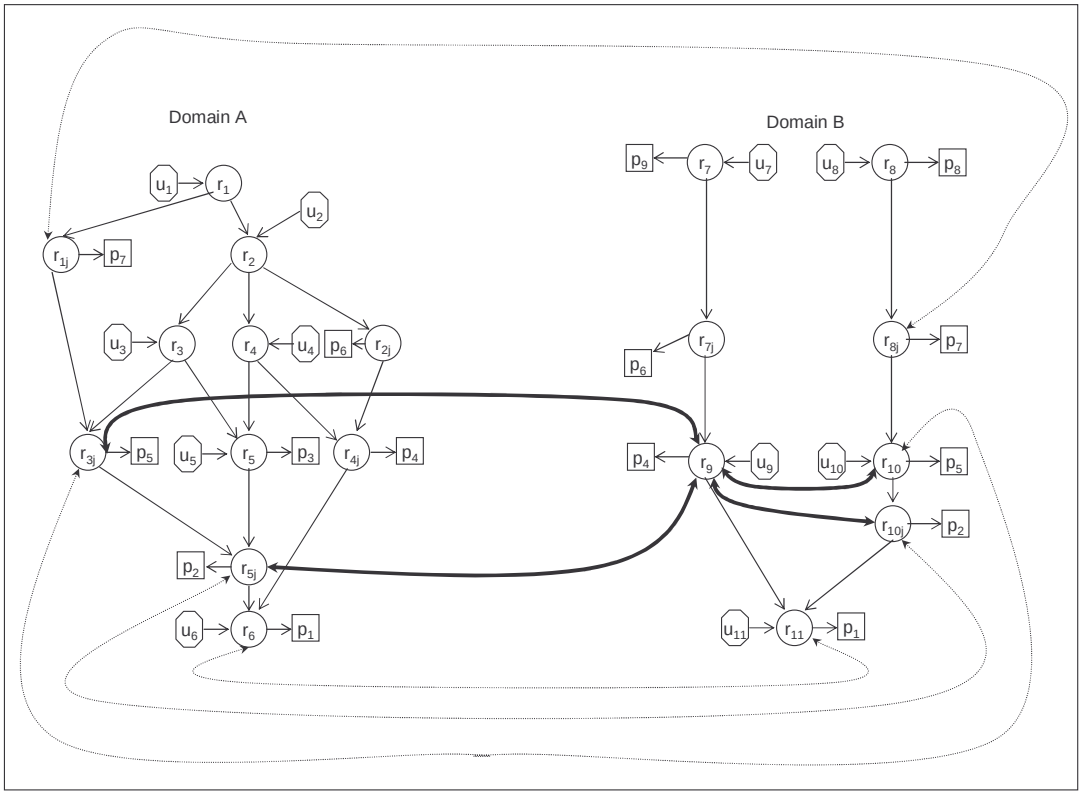

Figure 6. An integrated RBAC policy after applying procedure role-integrate

4.4 IP formulation of a multi-domain RBAC policy

In the following, we describe a procedure for formulating multi-domain policy integration problem into an integer program (IP) [Wol98]. In the IP formulation of RBAC policy, all the constraints such as hierarchical, SoD, permitted and restricted access constraints are defined using linear equations. The variables used in these equations convey both user and role information. For instance, the variables are of the form u_{ir_j}, where the first subscript *i* identifies the user and the second subscript r_j specifies the role. The variable u_{ir_j} is a binary variable, *i.e.*, it can take a value of '0' or '1'. If the variable $u_{ir_j} = 1$ then user u_i is authorized for role r_j, otherwise u_i is not authorized for r_j. If user u_i and role r_j are from different domains and $u_{ir_j} = 0$ then there should not be any path from the user node u_i to the role node r_j. Note that the given RBAC policy may be inconsistent and there may exist a path between user u_i from one domain and role r_j from another domain, and in the solution to the IP problem $u_{ir_j} = 0$. This inconsistency is resolved by dropping an inter-domain arc that lies in the path between the user node u_i and role node r_j.

To reduce the number of constraint equations, only one user assigned to the senior most role of each role hierarchy is considered. Note that there can be multiple role hierarchies in a single domain. In such a case, all the users assigned to the senior most roles of their corresponding hierarchies are considered. For instance, in figure 7, domain B has two role hierarchies with senior most roles r_7 and r_8. In the integer programming formulation shown in figure 8, both u_7 and u_8 are considered and all the users that are assigned the junior roles are omitted. In case some user, say u_i, is assigned the senior most roles of two or more role hierarchies, a new role is created and is made senior to all the senior most roles of such hierarchies. User u_i is then assigned the newly created senior role.

4.4.1 Constraint transformation rules

We use the following transformation rules to generate IP equations for an RBAC policy.

Intra-domain constraints

1. In the graph model, a user to role assignment is specified by an arc from user node to role node. In algebraic form this can be written as the following equation:

   $u_{ir_j} = 1$, where user u_i is assigned role r_j.

2. In the graph model, if some user u_i is not authorized to access a role r_j then there does not exist a path from the user node u_i to the role node r_j. Note that path in this case includes both intra-domain and inter-domain links/arcs. This restricted access can be specified in equation form by setting u_{ir_j} to 0.

3. If a role r_k has multiple parents, say r_1, r_2,…, r_n, and a user u_i is authorized for one or more of the parent roles of r_k, then u_i is also authorized for role r_k. This can be stated in the following equation form:

$$\sum_{j=1,\, j\neq k}^{n} u_{ir_j} - u_{ir_k} \geq 0$$

$$u_{ir_j} - u_{ir_k} \leq 0, \quad \text{for } 1 \leq j \leq n$$

If the role r_k has only one parent, say r_j, then the above equations can be reduced into the following equation:

$$u_{ir_j} - u_{ir_k} = 0$$

4. In the graph-based model, SoD constraint between two conflicting roles r_j and r_k is represented by a double-headed arrow between roles r_j and r_k. In the IP formulation, this SoD constraint can be written as:

$$u_{ir_j} + u_{ir_k} = 1, \qquad \text{for each user } u_i \text{ authorized for } r_j \text{ or } r_k$$

Inter-domain constraints

5. Consider a cross domain arc from role r_j to role r_k with r_j and r_k belonging to different domains. Suppose a user u_i is authorized for role r_j and both user u_i and role r_j belong to the same domain. In this case, user u_i may also access the cross-domain role r_j. This can be captured by the following set of equations:

$$u_{ir_j} - u_{ir_k} \leq 1$$

$$u_{ir_j} - u_{ir_k} \geq 0$$

The inequality relation in the above equation leaves room for relaxing a cross-domain access constraint in case it conflicts with other constraints.

6. Consider two users u_i and u_j who are authorized for some role r_k. Let there be a cross-domain link from role r_k to role r_l. Suppose user u_i and role r_k belong to the same domain. If both u_i and u_j belong to the same domain and u_i is authorized for the cross-domain role r_l then u_j is also authorized for the role r_l. However if u_i and u_j belong to different domains, then u_j may not be able to access role r_l, because the cross-domain arc allowing u_j to access r_k may be dropped from the solution of IP problem. Formally:

$$\text{if } \text{domain}(u_i) = \text{domain}(u_j) = \text{domain}(r_j) \neq \text{domain}(r_k) \text{ and } u_{jr_k} = u_{ir_k} \text{ then}$$

$$\left(u_{ir_k} - u_{ir_l}\right) - \left(u_{jr_k} - u_{jr_l}\right) = 0$$

$$\text{if } \text{domain}(u_i) = \text{domain}(r_j) \neq \text{domain}(r_k) = \text{domain}(u_j) \text{ and } u_{jr_k} \leq u_{ir_k} \text{ then}$$

$$\left(u_{ir_k} - u_{ir_l}\right) - \left(u_{jr_k} - u_{jr_l}\right) \leq 2$$

$$\left(u_{ir_k} - u_{ir_l}\right) - \left(u_{jr_k} - u_{jr_l}\right) \geq 0$$

7. Inter-domain SoD constraint may exist between two cross-domain roles or between two role belonging to same domain but can be accessed by cross-domain users. In IP formulation, the cross-domain SoD constraint between two roles r_j and r_k is defined by the following constraint equation:

$$u_{ir_j} + u_{ir_k} \leq 1, \qquad \text{for all users } u_i \text{ such that } u_i \text{ can access either } r_i \text{ or } r_k$$

4.5 Optimality Criteria

Once the RBAC constraints are transformed into linear equations by using the above transformation rules, the multi-domain RBAC policy can be formulated as the following integer programming problem.

$$\begin{aligned} &\text{maximize} && c^T u \\ &\text{subject to} && Au \leq b \\ & && \forall u_{ir_j} \in u,\ u_{ir_j} = 0 \text{ or } 1 \end{aligned}$$

Where, c is the cost vector and A is the constraint matrix. The cost vector c defines the optimality criteria. The main purpose of formulating the multi-domain RBAC policy into an IP problem is to find a feasible solution (a set of permitted user to role accesses that do not violate the given security and autonomy requirements of individual domains) that maximizes the objective function according to given optimality criterion. One of the optimality criteria might be to maximize the number of cross domain role accesses. In this case the objective function is the sum of all variables defining inter-domain user to role accesses. Example 1 given below uses this optimality criterion.

Maximizing cinter-domain accesses may lead to relaxation or dropping of some of the administrator specified constraints which may not be desirable in certain situations. When administrative constraints are to be preserved, the element of cost vector corresponding to the administrator specified constraint is assigned a higher value. Example 2 given below addresses this issue.

The following examples use the multi-domain RBAC policy shown in figure 7. In this policy role r_2 is made senior to role r_{7j} by the administrator. This constraint is represented as a dashed arrow from role r_2 to r_{7j} in figure 7**.** Moreover, there are inter-domain links between roles r_{1j} and r_{8j}, r_{3j} and r_{10}, r_{5j} and r_{10j}, and r_6 and r_{11}. This is an inconsistent policy because user u_1 is allowed to access roles r_9, r_{10} and r_{10j}, which is a violation of SoD constraint defined between roles r_9 and r_{10} and r_9 and r_{10j}. The constraint equations for this multi-domain RBAC policy are shown in figure 8.

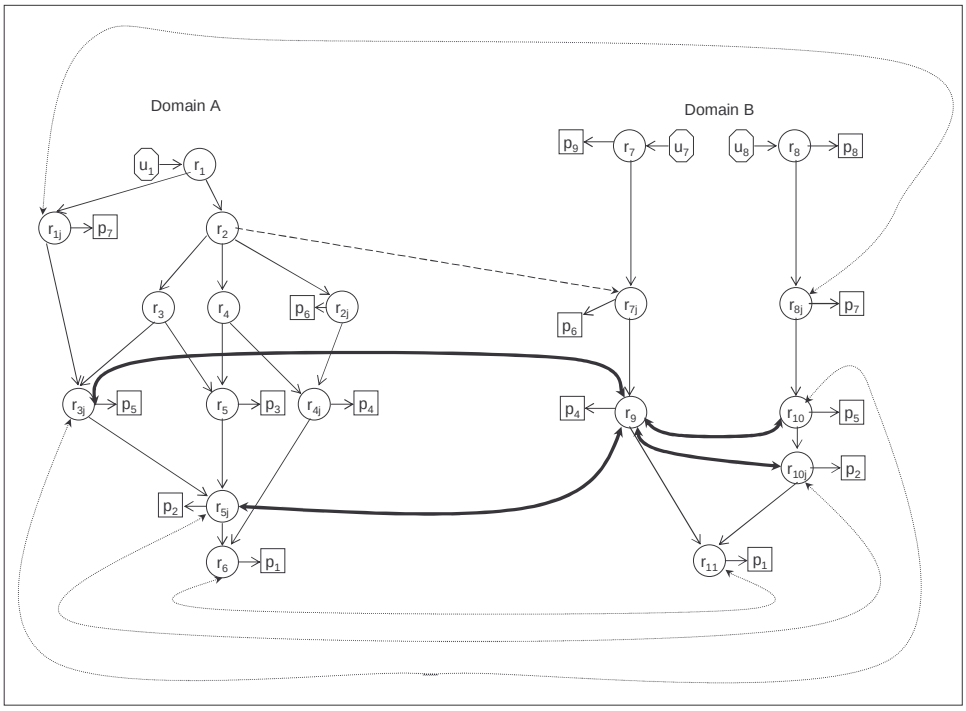

Figure 7. An insecure multi-domain RBAC Policy

Example 1

In this example, we will use the maximization of inter-domain user to role accesses as an optimality criterion and resolve the inconsistency present in the multi-domain RBAC policy shown in figure 7. The following objective function can be used for this optimality measure:

$$\text{maximize} \quad u_{1r_{7j}} + u_{1r_9} + u_{1r_{11}} + u_{1r_{8j}} + u_{1r_{10}} + u_{1r_{10j}} + u_{8r_{1j}} + u_{8r_{3j}} + u_{8r_{5j}} + u_{8r_6}$$

A feasible solution that maximizes the above objective function has the following values for the given inter-domain variables:

$$u_{1r_{7j}} = 0,\ u_{1r_9} = 0,\ u_{1r_{11}} = u_{1r_{8j}} = u_{1r_{10}} = u_{1r_{10j}} = u_{8r_{1j}} = u_{8r_{3j}} = u_{8r_{5j}} = u_{8r_6} = 1$$

Since, $u_{1r_{7j}} = 0$ and $u_{1r_9} = 0$, therefore the administrator specified inter-domain arc from role r_2 to role r_{7j} should be removed. The resulting RBAC policy is similar to the integrated policy shown in figure 6.

Domain A
Hierarchical Constraints
$u_{1r_1} = 1$
$u_{1r_1} - u_{1r_2} = 0, \quad u_{1r_2} - u_{1r_{2j}} = 0, \quad u_{1r_{2j}} - u_{1r_{4j}} = 0, \quad u_{1r_{4j}} - u_{1r_6} = 0$
$u_{1r_1} - u_{1r_{1j}} = 0, \quad u_{1r_{1j}} - u_{1r_{3j}} = 0, \quad u_{1r_{3j}} - u_{1r_{5j}} = 0, \quad u_{1r_{5j}} - u_{1r_6} = 0$
$u_{1r_2} - u_{1r_4} = 0, \quad u_{1r_4} - u_{1r_5} = 0, \quad u_{1r_5} - u_{1r_{5j}} = 0$
$u_{1r_2} - u_{1r_3} = 0, \quad u_{1r_3} - u_{1r_5} = 0, \quad u_{1r_3} - u_{1r_{3j}} = 0$
Domain B
Hierarchical Constraints
$u_{7r_7} = 1, \quad u_{8r_8} = 1$
$u_{7r_7} - u_{7r_{7j}} = 0, \quad u_{7r_{7j}} - u_{7r_9} = 0, \quad u_{7r_9} + u_{7r_{10j}} - u_{7r_{11}} = 0$
$u_{8r_8} - u_{8r_{8j}} = 0, \quad u_{8r_{8j}} - u_{8r_{10}} = 0, \quad u_{8r_{10}} - u_{8r_{10j}} = 0, \quad u_{8r_9} + u_{8r_{10j}} - u_{8r_{11}} = 0$
$u_{7r_8} = u_{7r_{8j}} = u_{7r_{10}} = u_{7r_{10j}} = 0$
$u_{8r_7} = u_{8r_{7j}} = u_{8r_9} = 0$
SOD
$u_{7r_9} + u_{7r_{10}} = 1, \quad u_{7r_9} + u_{7r_{10}j} = 1, \quad u_{8r_9} + u_{8r_{10}} = 1, \quad u_{8r_9} + u_{8r_{10}j} = 1$
Inter-domain Constraints
links
$u_{1r_{1j}} - u_{1r_{8j}} \le 1, \quad u_{1r_{1j}} - u_{1r_{8j}} \ge 0, \quad u_{1r_{8j}} - u_{1r_{10}} = 0 \quad u_{1r_{10}} - u_{1r_{10j}} = 0,$
$u_{1r_{10j}} + u_{1r_9} - u_{1r_{11}} = 0$
$u_{1r_{3j}} - u_{1r_{10}} \le 1, \quad u_{1r_{3j}} - u_{1r_{10}} \ge 0, \quad u_{1r_{5j}} - u_{1r_{10j}} \le 1,$
$u_{1r_{5j}} - u_{1r_{10j}} \ge 0, \quad u_{1r_6} - u_{1r_{11}} \le 1, \quad u_{1r_6} - u_{1r_{11}} \ge 0,$
$(u_{1r_{10}} - u_{1r_{3j}}) - (u_{8r_{10}} - u_{8r_{3j}}) \le 0, \quad (u_{1r_{10j}} - u_{1r_{5j}}) - (u_{1r_{10j}} - u_{8r_{5j}}) \le 0,$
$(u_{1r_{11}} - u_{1r_6}) - (u_{8r_{11}} - u_{8r_6}) \le 0, \quad (u_{1r_{11}} - u_{1r_6}) - (u_{7r_{11}} - u_{7r_6}) \le 0$
$u_{8r_{8j}} - u_{8r_{1j}} \le 1, \quad u_{8r_{8j}} - u_{8r_{1j}} \ge 0, \quad u_{8r_{1j}} - u_{8r_{3j}} = 0, \quad u_{8r_{3j}} - u_{8r_{5j}} = 0,$
$u_{8r_{5j}} - u_{8r_6} = 0, \quad u_{8r_{10}} - u_{8r_{3j}} \le 1, \quad u_{8r_{10}} - u_{8r_{3j}} \ge 0, \quad u_{8r_{10j}} - u_{8r_{5j}} \le 1,$
$u_{8r_{10j}} - u_{8r_{5j}} \ge 0, \quad u_{8r_{11}} - u_{8r_6} \le 1, \quad u_{8r_{11}} - u_{8r_6} \ge 0$
$(u_{8r_{3j}} - u_{8r_{10}}) - (u_{1r_{3j}} - u_{1r_{10}}) \le 0, \quad (u_{8r_{5j}} - u_{8r_{10j}}) - (u_{1r_{5j}} - u_{1r_{10j}}) \le 0,$
$(u_{8r_6} - u_{8r_{11}}) - (u_{1r_6} - u_{1r_{11}}) \le 0$
$u_{1r_8} = u_{1r_7} = 0, \quad u_{8r_1} = u_{8r_2} = u_{8r_{2j}} = u_{8r_3} = u_{8r_4} = u_{8r_5} = u_{8r_{4j}} = 0,$
$u_{7r_1} = u_{7r_1j} = u_{7r_2} = u_{7r_{2j}} = u_{7r_3} = u_{7r_{3j}} = u_{7r_4} = u_{7r_{4j}} = u_{7r_5} = u_{7r_{5j}} = 0$
Inter-domain SOD
$u_{1r_{3j}} + u_{1r_9} \le 1, \quad u_{1r_{5j}} + u_{1r_9} \le 1, \quad u_{1r_9} + u_{1r_{10}} \le 1, \quad u_{1r_9} + u_{1r_{10j}} \le 1$
$u_{8r_{3j}} + u_{8r_9} \le 1, \quad u_{8r_{5j}} + u_{8r_9} \le 1$

Figure 8. Constraint equations for the multi-domain RBAC policy shown in figure 7.

Example 2

Suppose the optimality criterion is to maximize the inter-domain accesses specified by the administrator. In this case the objective function can be written as:

$$\text{maximize} \quad c_A(u_{1r_{7j}} + u_{1r_9} + u_{1r_{11}}) + c_l(u_{1r_{8j}} + u_{1r_{10}} + u_{1r_{10j}} + u_{1r_{11}} + u_{8r_{1j}} + u_{8r_{3j}} + u_{8r_{5j}} + u_{8r_6})$$

where, $c_A > \frac{8}{3} c_l$.

A feasible solution that maximizes the above objective function has the following values for the inter-domain variables:

$$u_{1r_{7j}} = 1,\ u_{1r_9} = 1,\ u_{1r_{11}} = 1, u_{1r_{8j}} = 0,\ u_{1r_{10}} = 0,\ u_{1r_{10j}} = 0,\ u_{8r_{1j}} = 1,\ u_{8r_{3j}} = 1,\ u_{8r_{5j}} = 1,\ u_{8r_6} = 1$$

Since user u_1 cannot access role r_{8j}, r_{10}, and r_{10j}, therefore the following inter-domain arcs from r_{1j} to r_{8j}, r_{3j} to r_{10}, and r_{5j} to r_{10j} can be dropped. The corresponding multi-domain policy is shown in figure 9. Note that the link arrows (r_{8j}- r_{1j}), (r_{10}- r_{3j}), and (r_{10j}- r_{5j}) are unidirectional.

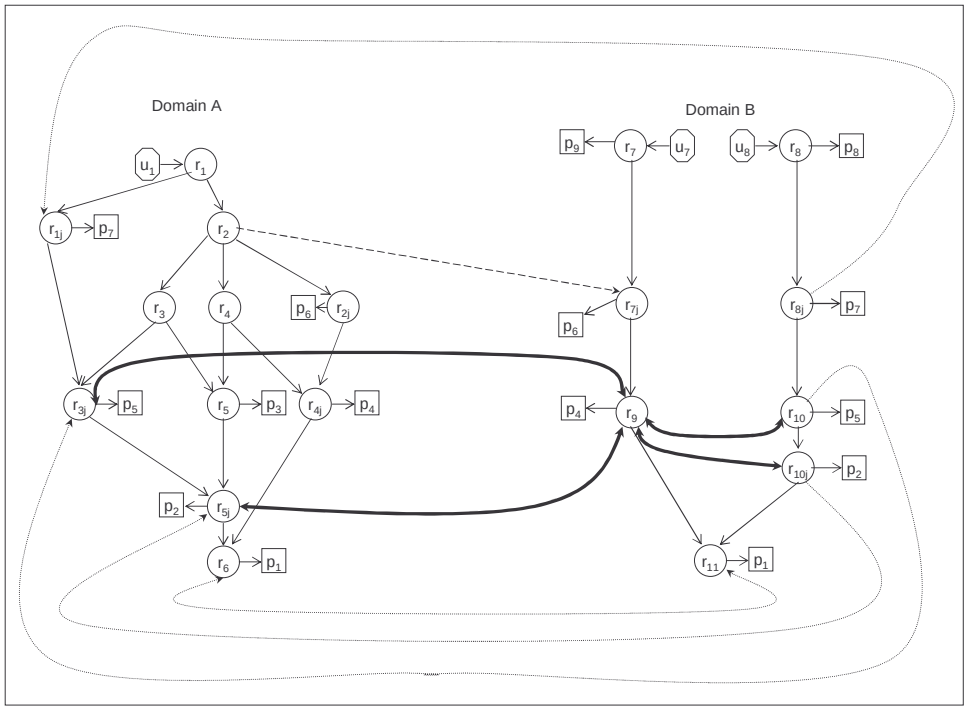

Figure 9. An optimal RBAC policy that retains administrator specified role access constraints.

5 Conclusion and future work

In this paper, we discuss the issue of secure interoperation in a multi-domain environment. In particular, we present a policy integration mechanism for integrating RBAC policies of multiple domains. We consider RBAC model because of its inherent richness in expressing a wide variety of constraints that exist in most commercial systems. The proposed policy integration mechanism described in this paper, consists of two phases. In the first phase, a global security policy is generated from the security policies of individual domains. However, this policy may consist of a set of conflicting constraints that make it inconsistent. These conflicts are resolved by relaxing some of the inter-domain access constraints. However, it is ensured that relaxation of any inter-domain constraint does not cause any violation of the security and autonomy requirements of the constituent domains. For resolving conflicts in a multi-domain policy, we propose an integer programming-based approach that determines an optimal set of allowable accesses. Although in the examples only two domains are considered for policy integration, the mechanism described in this paper is generic enough and can work for any number of domains provided the domains' security policies are expressed in RBAC framework. For a multi-domain environment consisting of more than two domains, a pair wise approach can be used to generate the global security policy. For instance, one pair-

wise merging strategy is to first integrate the security policies of any two domains and then integrate the resulting policy with the third domain's policy. This process continues until all policies are integrated.

In this paper, we considered monotype inheritance hierarchy only. However, RBAC model supports multiple types of hierarchies. As a future work, we are planning to extend the current policy integration mechanism in order to incorporate various types of hierarchies. Moreover, we would also like to investigate the problem of secure interoperation in presence of temporal constraints.

6 References

[Ber99] E. Bertino, E. Ferrari, and V. Atluri, "The Specification and Enforcement of Authorization Constraints in Workflow Management Systems," *ACM Transactions on Information and System Security*, 2(1):65-104, 1999.

[Bel73] D. Bell and L. Lapadula, "Secure Computer Systems: Mathematical Foundations," Technical Report MTR-2547, Vol. 1, MITRE Corporation, March 1973.

[Bib77] K. Biba, "Integrity Considerations for Secure Computer Systems," Technical Report MTR-3153, Vol. 1, MITRE Corporation, April 1977.

[Bon96] P.A. Bonatti, M. L. Sapino, V.S. Subrahmanian, "Merging Heterogeneous Security Orderings," *ESORICS 1996*, pp. 183-197

[Fer01] D. Ferraiolo, R. Sandhu, S. Gavrila, R. Kuhn, R. Chandramouli, "The NIST Model for Role-Based Access Control: Towards a Unified Standard," *ACM Transactions on Information and System Security*, Vol. 4, Issue 3, August 2001, pp. 224-274.

[Gav98] S. I. Gavrila , J. F. Barkley, "Formal Specification for Role Based Access Control User/role and Role/role Relationship Management," *Proceedings of the third ACM workshop on Role-based access control*, Fairfax, Virginia, United States, pp. 81-90, October 1998.

[Gon96] L. Gong and X. Qian, "Computational Issues in Secure Interoperation", *IEEE Transaction on Software and Engineering*, Vol. 22, No. 1, January 1996.

[Gua02] G. Yan, W. K. Ng, E. Lim, "Product Schema Integration for Electronic Commerce - A Synonym Comparison Approach," *IEEE TKDE* Vol. 14, No. 3 pp. 583-598, June 2002.

[Har76] M. Harrisson, W. Ruzzo, and J. Ullman, "Protection in Operating Systems," *Communications of the ACM*, Vol. 19, No. 2, August 1976, pp. 461-471.

[Hos91] H. Hosmer, "Metapolicies I," *ACM SIGSAC Review*, 1992, pp. 18-43.

[Jos01a] J. B. D. Joshi, W. G. Aref, A. Ghafoor and E. H. Spafford, "Security Models for Web-based Applications," *Communications of the ACM*, Vol. 44, No. 2, Feb. 2001, pages 38-72.

[Jos01b] J. B. D. Joshi, A. Ghafoor, W. Aref, E. H. Spafford, "Digital Government Security Infrastructure Design Challenges", *IEEE Computer*, Vol. 34, No. 2, February 2001, pages 66-72.

[Jos02] J. B. D. Joshi, E. Bertino, A. Ghafoor, "Temporal Hierarchies and Inheritance Semantics for GTRBAC," *Seventh ACM Symposium on Access Control Models and Technologies*, pp. 74-83, June 2002.

[Koc02] M. Koch, L.V. Mancini and F. P. Presicce, "A Graph-Based Formalism for RBAC," *ACM Transactions on Information and System Security*, Vol. 5, No. 3, pp. 332-365, August 2002.

[Li94] W. S. Li and C. Clifton, "Semantic Integration in Heterogeneous Databases Using Neural Networks," *VLDB* 1994.

[Osb00] S. L. Osborn, R. Sandhu, Q. Munawer, "Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies," *ACM Transactions on Information and System Security*, Vol. 3, No. 2, February 2000, pp. 85-106.

[Osb02] S. L. Osborn, “Integrating Role Graphs: A Tool for Security Integration,” Data and Knowledge Engineering, Vol. 43 No. 3, pp. 317-333, 2002.

[Pow00] R. Power, “"Tangled Web": Tales of Digital Crime from the Shadows of Cyberspace,” *Que/Macmillan Publishing*, Aug. 31, 2000.

[San96] R. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, “Role Based Access Control Models”, *IEEE Computer* Vol. 29, No 2, February 1996

[Vet98] V. Vet and N. Mars “Bottom-Up Construction of Ontologies,” *IEEE TKDE*, Vol. 10, No. 4, pp. 513-526, August 1998.

[Wol98] L. A. Wolsey, *Integer Programming*, John Wiley, New York, 1998.