

The A-IRBAC 2000 Model: Administrative Interoperable Role-Based Access Control

Jalal Al-Muhtadi, Apu Kapadia, Roy Campbell, Dennis Mickunas
{almuhtad | akapadia | rhc | mickunas}@uiuc.edu

Department of Computer Science
University of Illinois at Urbana Champaign

Abstract

Secure interaction and interoperability between administrative domains is a major concern. For domains that utilize RBAC, Kapadia et al. [Kap00] proposed the IRBAC 2000 model, which can be used to establish a flexible policy for dynamic inter-domain role translations. However, there are scenarios under which the IRBAC model becomes difficult to manage. One such scenario is where interoperability is desired among large and complex role hierarchies. Another is where there is a desire to distribute the administrative authority for managing inter-domain translations between different security officers. Therefore, we introduce the Administrative IRBAC 2000 model. The A-IRBAC builds over IRBAC model by employing RBAC to manage inter-domain role translations, following the concepts outlined in ARBAC97 [San99] by Sandhu et al.

1. Introduction

Throughout this paper, we use the term *domain* to refer to an *administrative* domain, which is defined as “A collection of hosts and routers, and the interconnecting network(s), managed by a single administrative authority [Mal96].” This single administrative authority will include one or more *senior security officers*.

Secure interoperability between two domains that utilize the RBAC (*role-based access control*) model is a major concern. We have proposed [Kap00] a policy framework that facilitates the secure interoperability between two RBAC-based domains through the use of dynamic role translation. This model is called the

Interoperable Role-Based Access Control (IRBAC 2000).

In situations where interoperability is desired among many different domains with significantly different role hierarchies, managing dynamic translations between the local role hierarchy and all the foreign hierarchies becomes tedious. Our aim is to introduce a flexible and dynamic method to distribute and simplify the management of role translations without violating domain-wide security policies. Inspired by the ARBAC model proposed by Sandhu et al. in [San97], [San99] and [San99b], we present the *Administrative IRBAC 2000 model* or A-IRBAC. A-IRBAC utilizes RBAC to administer and manage dynamic role translations between the local role hierarchy and foreign hierarchies. This model also provides the senior security officer with additional security tools for enforcing the organization’s global security policies, limiting junior security officers’ authority and auditing events for security and accountability purposes. This model can be combined with ARBAC97 [San99] or can be used independently.

The remainder of this paper is organized as follows. Section 2 provides some background information about the dynamic role translation model, IRBAC 2000. Section 3 explains the A-IRBAC 2000 and its components in detail. Section 4 describes our implementation of A-IRBAC. Section 5 talks about future work.

2. The IRBAC 2000 Model

In [Kap00] we proposed *IRBAC 2000*, a policy framework that facilitates the secure interoperability between two domains. This

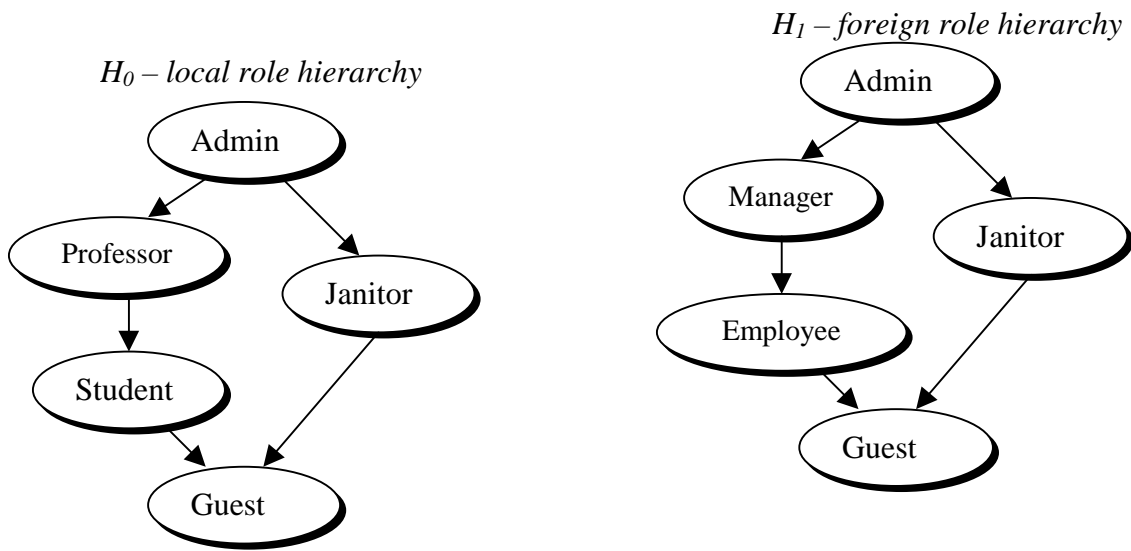


Figure 1: A local and a foreign role hierarchy.

policy framework works by defining a set of associations between the local and foreign role hierarchies, which forms a combined hierarchy that is partially ordered. This partial ordering is used to enable interoperation between the two domains while attaining the level of flexibility desired.

To formalize this, let R_0 denote the set of roles in the local domain D_0 . Let R_1 denote the set of roles in some foreign domain D_1 . Let the hierarchies H_0 and H_1 be partial orderings on R_0 and R_1 respectively. We define $x \geq y$ to mean that role x is *senior to* role y (or y is *junior to* x). This relation implies that role x inherits role y 's permissions. Further, members in role x are

implicitly members in role y . Roles with a subscript, like $Professor_{R_0}$, will read as $Professor$ from R_0 . Let R_1R_0 denote the dynamic role translation from the foreign domain with roles R_1 to the local domain with roles R_0 , note that $R_1R_0 \subseteq R_1 \times R_0$. To illustrate this, we give a simple example. In Figure 1, we have two role hierarchies: a local hierarchy (H_0) and a foreign hierarchy (H_1). Although the structures of the role Hierarchies are similar, they differ in their semantics. Note that both domains have a "Guest Role". If a foreign role is not understood, it can be mapped to the Guest Role. However, if a foreign object with the $Manager_{R_1}$ role wants to interoperate with an appli-

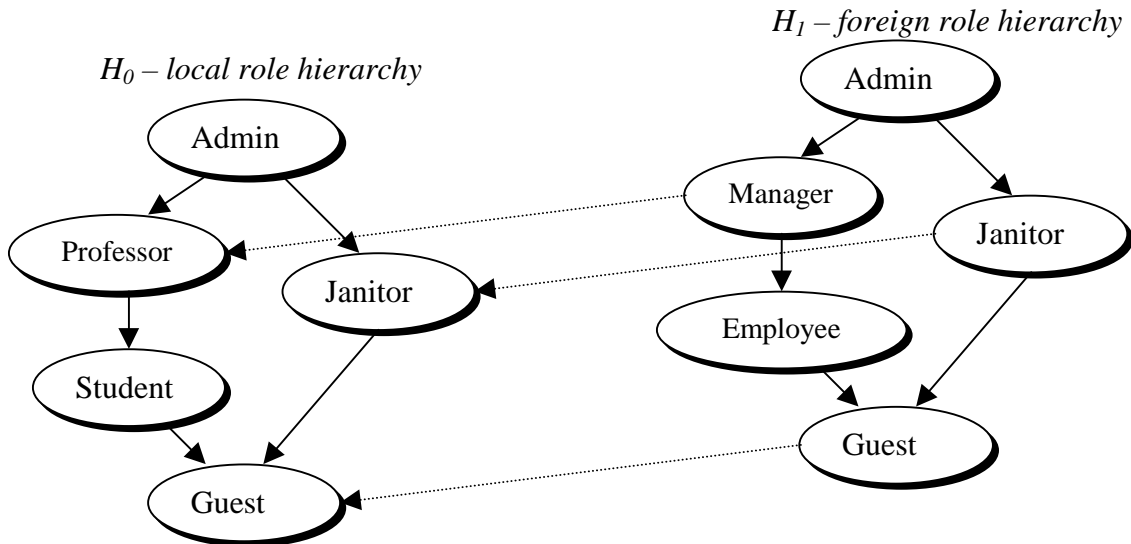


Figure 2: Partial ordering between the two hierarchies.

cation within the local domain that usually allows only local Professor roles, and if the security policies in the local domain permit a Manager_{R_I} to have the local “Professor” role, then the application must be able to translate the Manager_{R_I} role into something meaningful. To accomplish this, IRBAC creates a partial ordering using the two role hierarchies (Figure 2). By doing so, one can easily manipulate the levels of access for specific foreign roles. For instance, the dashed arrow from Manager_{R_I} role in H_I to Professor_{R_O} role in R_0 implies that R_I ’s “Manager” is translated into R_0 ’s “Professor”. Let us denote this as $\text{Manager}_{R_I} \mapsto \text{Professor}_{R_0}$. Where “ \mapsto ” reads “maps to”. This is equivalent to writing: $(\text{Manager}, \text{Professor}) \in R_I R_0$. Note that Admin_{R_I} role is senior to Manager_{R_I}, therefore, Admin_{R_I} can utilize the Manager_{R_I} role translation, thus, $(\text{Admin}, \text{Professor}) \in R_I R_0$ implicitly. Therefore, the “ \mapsto ” relation is transitive. From Figure 2, $R_I R_0$ consists of the following:

$R_I R_0 = \{ (\text{Manager}, \text{Professor}), (\text{Admin}, \text{Professor}), (\text{Janitor}, \text{Janitor}), (\text{Admin}, \text{Janitor}), (\text{Guest}, \text{Guest}), (\text{Employee}, \text{Guest}), (\text{Janitor}, \text{Guest}), (\text{Manager}, \text{Guest}), (\text{Admin}, \text{Guest}) \}$.

To provide further flexibility for the interoperability model, *non-transitive* associations are introduced between roles. For example, in Figure 3, the association between Manager_{R_I} and Professor_{R_O} illustrated by the bold dashed arrow with an “NT” label is a *non-*

transitive association. This is denoted as $\text{Manager}_{R_I} \mapsto_{NT} \text{Professor}_{R_0}$. A non-transitive association prevents implicit mappings from the foreign domain. In this case, $(\text{Manager}, \text{Professor}) \in R_I R_0$, but $(\text{Admin}, \text{Professor}) \notin R_I R_0$. Thus, in Figure 3, $R_I R_0 = \{ (\text{Manager}, \text{Professor}), (\text{Janitor}, \text{Janitor}), (\text{Admin}, \text{Janitor}), (\text{Guest}, \text{Guest}), (\text{Employee}, \text{Guest}), (\text{Janitor}, \text{Guest}), (\text{Manager}, \text{Guest}), (\text{Admin}, \text{Guest}) \}$.

3. The A-IRBAC Model

Figure 4 shows the main components in A-IRBAC 2000. Users and permissions are assigned to local roles R_0 . There is a partial ordering relationship among the local roles. This relation forms the role hierarchy H_0 . The local domain should be able to interoperate securely with foreign domains using the IRBAC 2000 model. For example, in Figure 4, the local hierarchy has established associations with foreign hierarchies H_1, \dots, H_n . In addition to the regular roles and permissions, A-IRBAC introduces a separate set of roles and permissions, namely, the *interoperability administrative roles* (IAR) and permissions (IAP). IAP and users are assigned to IAR. A hierarchy of roles is formed by members of IAR (H_{IAR}). This proposed hierarchy can either be the same hierarchy that is used in ARBAC97 [San99] for administering RBAC, or it can be a separate administrative hierarchy in which the sole re-

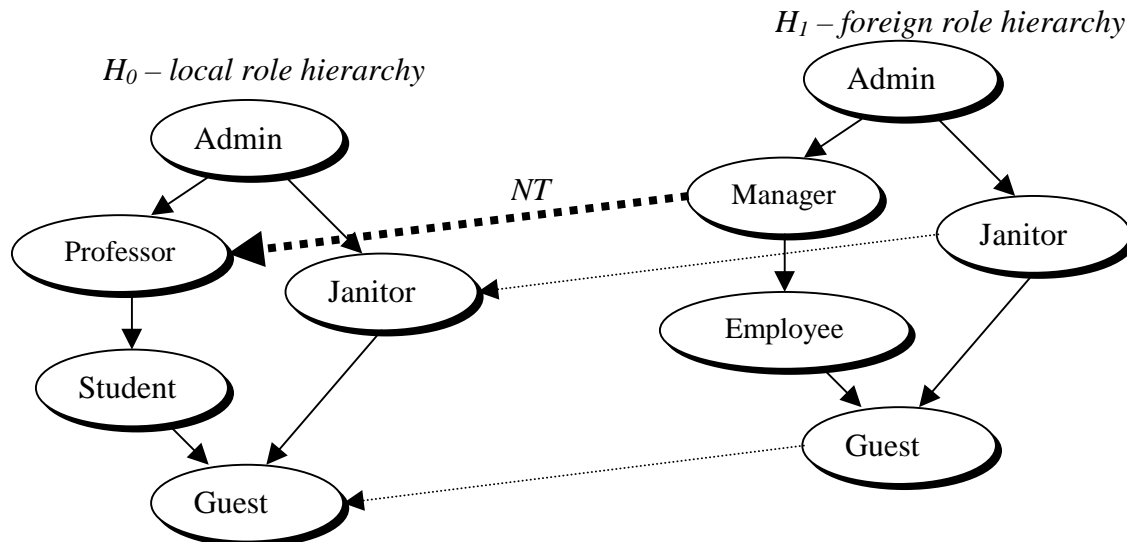


Figure 3: transitive translation.

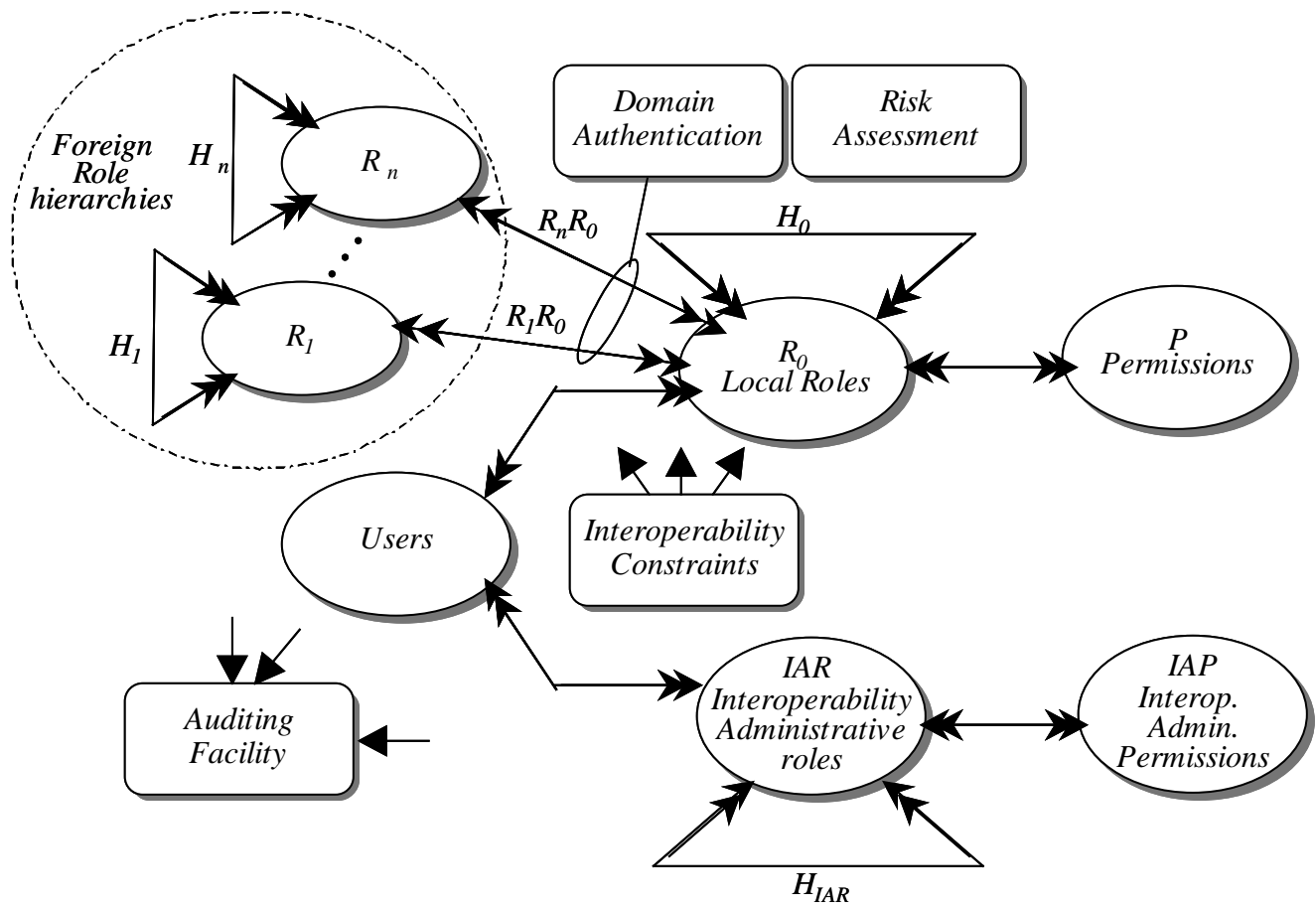


Figure 4: A-IRBAC 2000 Components

sponsibility of roles is to assign and revoke associations between foreign roles and local roles.

Administering role translations involves controlling which foreign roles can be mapped to which local roles, establishing inter-domain translations, revoking these translations and setting up interoperability constraints and conditions. As for assigning users and permissions to regular roles and IAR this could be carried out by the single security authority of the domain (as with conventional RBAC management) or by employing ARBAC97 [San99]. The rest of the components of A-IRBAC are described in more detail in the following subsections.

3.1 Domain Authentication Component

To enable secure interoperation with other domains, some mechanism for authenticating foreign roles and their originating domain should exist. For instance, the local domain may per-

mit the role *Employee_{R1}* from domain *D₁* to be mapped to some local role, but how could the local domain confirm that the foreign role actually came from domain *D₁*? The authentication component filters out foreign principals with phony roles.

One convenient way to accomplish this is by embedding a certificate within each foreign principal. This certificate indicates the role name(s) and the originating domain of the principal. A trusted third party Certificate Authority (CA) should sign these certificates and validate them.

3.2 Interoperability Constraints Component

The “Interoperability Constraints” component allows the senior security officers to specify system-wide constraints for interoperability. The global security policy of the organization is enforced with the help of these constraints. The concept behind this is similar to ARBAC97

constraints [San99]. For instance, the interoperability constraints can designate particular foreign domains as unsafe, thus prohibiting any kind of interoperability with these domains. The constraints can also prohibit foreign roles from translating into “sensitive” roles, such as roles in the payroll department. The constraints specified here override other settings in the A-IRBAC model.

3.3 Creating Role Translations

Creating role translations is similar to the way ARBAC97 [San99] assigns users or permissions to roles. First, some definitions are needed. Given roles x and y , where $x \leq y$, we define a role range as $[x, y] = \{r \in R_0 \mid x \leq r \leq y\}$. (This is merely a notational convenience for representing a contiguous set of roles.)

To be able to restrict the allowed translations per security officer, the concept of a prerequisite condition similar to that of ARBAC [San99] is adapted. The prerequisite condition is a Boolean expression that consists of a combination of the Boolean operators \wedge , \vee and \neg (for unary Not) and two predefined Boolean functions described below.

- $In_domain(D_i)$ evaluates to “true” if the foreign role in question belongs to domain D_i . False otherwise.
- $Mapped_to(x)$ where $x \in R_0$, evaluates to true if the foreign role in question can be translated (either directly or indirectly) to the local role x or to a local role senior to x . False otherwise.

Examples for using prerequisite conditions are provided below.

We define “ $can_assignT$ ” relation for assigning translations from a foreign domain to the domestic one. Formally:

$$can_assignT \subseteq IAR \times CR \times 2^{R_0}$$

where CR represents the set of all possible prerequisite conditions.

For example, $can_assignT(x, pc, [a, c])$ means that a member of the interoperability administrative role (IAR) x (or a role senior to it) can translate a foreign role that satisfies the prerequisite condition pc to any local role within the range $[a, c]$. The range $[a, c]$ is referred to as the *authority range*. A more detailed practical example is illustrated in the next paragraph.

Consider the local role hierarchy H_0 in the local domain D_0 shown in Figure 5a. The hierarchy here depicts a university “software research” environment. At the top of the hierarchy the Primary Investigator (PI) role exists (other roles could exist above this). Two research projects, Project 1 and Project 2, are undertaken in parallel. Each research project consists of the following roles: the project leader roles (PL1 and PL2), Researcher roles (RS1 and RS2), Senior Engineers (SE1 and SE2), and Programmer roles (Prog1 and Prog2). SRG is the Software Research Group role. Further, the hierarchy has a Guest role to which all foreign roles default. Figure 5b shows the administrative role hierarchy for managing interoperability (H_{IAR}). This hierarchy contains four administrative roles: the senior security officer role (SSO), Project 1 security officer role (SO1), Project 2 security officer role (SO2), and finally the SRG security officer role (SRGSO). Table 1 defines the values for the $can_assignT$ relation for this scenario.

Table 1: $can_assignT$ relation

IAR	Prerequisite condition	Authority range
SRGSO	Mapped_to(Guest)	{SRG}
SO1	$\neg In_domain(XYZ) \wedge \neg mapped_to(Prog2)$	[Prog1, PL1]
SO2	$\neg In_domain(foo) \wedge \neg mapped_to(Prog1)$	[Prog2, PL2]
SSO	\emptyset	[SRG, PI] \cup {Guest}

H_0 – Local role hierarchy

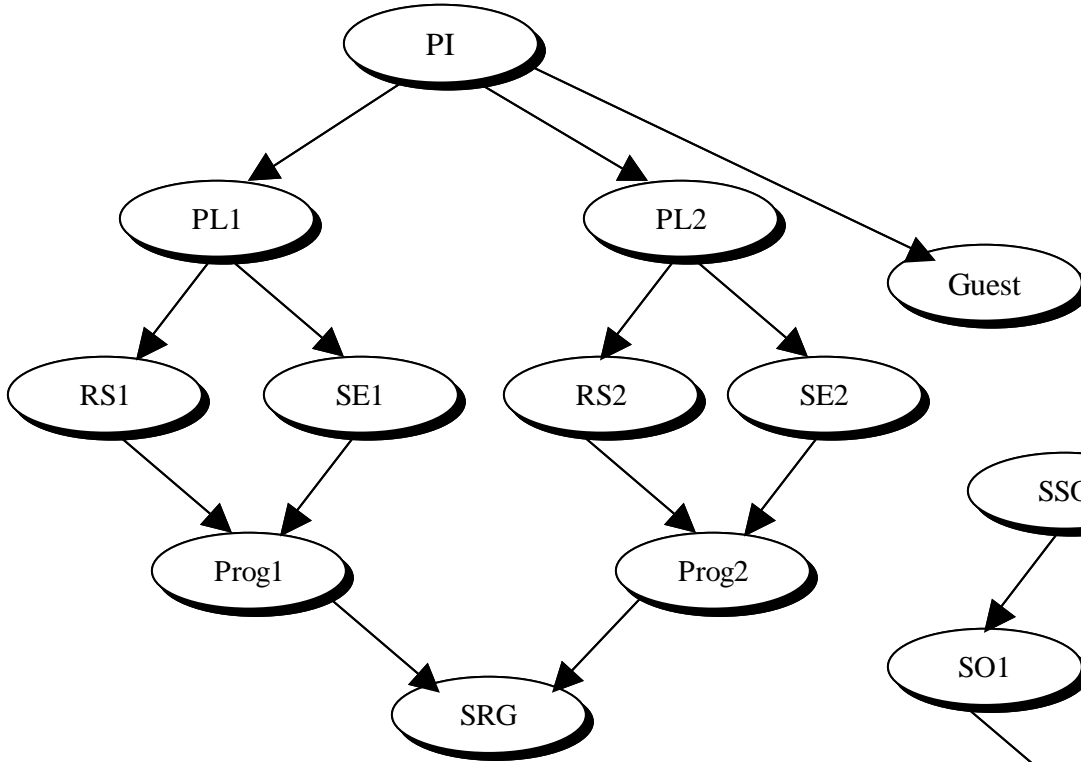


Figure 5a: Sample hierarchy.

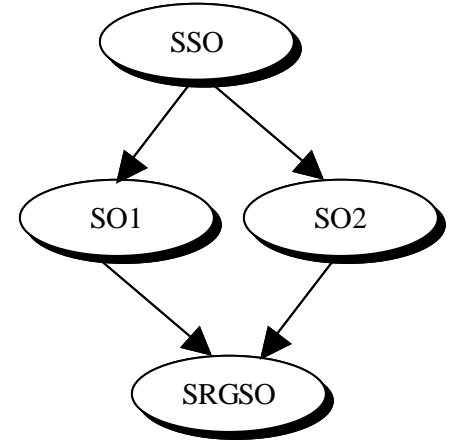


Figure 5b: H_{IAR}

In Table 1, the SRGSO role can translate any foreign role that has a mapping to role “Guest_{R0}” to role SRG_{R0}. However, if the foreign role does not translate to “Guest_{R0}” role, then SRGSO cannot translate it to SRG_{R0} as the prerequisite condition fails in this case. Here, we can assume that Guest_{R0} is a local role with limited permissions to which all foreign roles from selected domains are translated by default. Let us take a look at the SO1 role entry. The security officer for Project 1 has interoperability authority over all roles ranging between Prog1 and PL1 inclusively. Say, that we want to disallow any role translations from domain XYZ (who are researching a competitive product) to Project 1 roles in the local domain, so we specify “ $\neg In_domain(XYZ)$ ” in the prerequisite condition to limit SO1 authority. Note that we are enforcing another policy in the prerequisite condition, which ensures that a foreign role can never be mapped to roles in Project 1 and Project 2 at the same time. Like-

wise, the security officer for Project 2 cannot translate roles from domain “foo” to Project 2 roles. Note that evaluating a prerequisite and modifying the translation should be implemented as an atomic operation to prevent interferences while creating role translations. Since both administrative roles SO1 and SO2 are senior to SRGSO (Figure 5b), therefore, they will inherit SRGSO’s permissions. Hence, SO1 and SO2 can map any foreign role (that is mapped to Guest_{R0}) to SRG_{R0}. As for the SSO, no prerequisite condition is required, this is denoted by the symbol ‘ \emptyset ’.

To illustrate how global constraints could come in handy, say the senior security officer decides to disallow any translations from some untrusted domain, D_u . Hence, instead of modifying the prerequisite conditions for all administrative roles, it is enough to append the predicate “ $\neg In_domain(D_u)$ ” in the Interoperability Constraints Component described earlier.

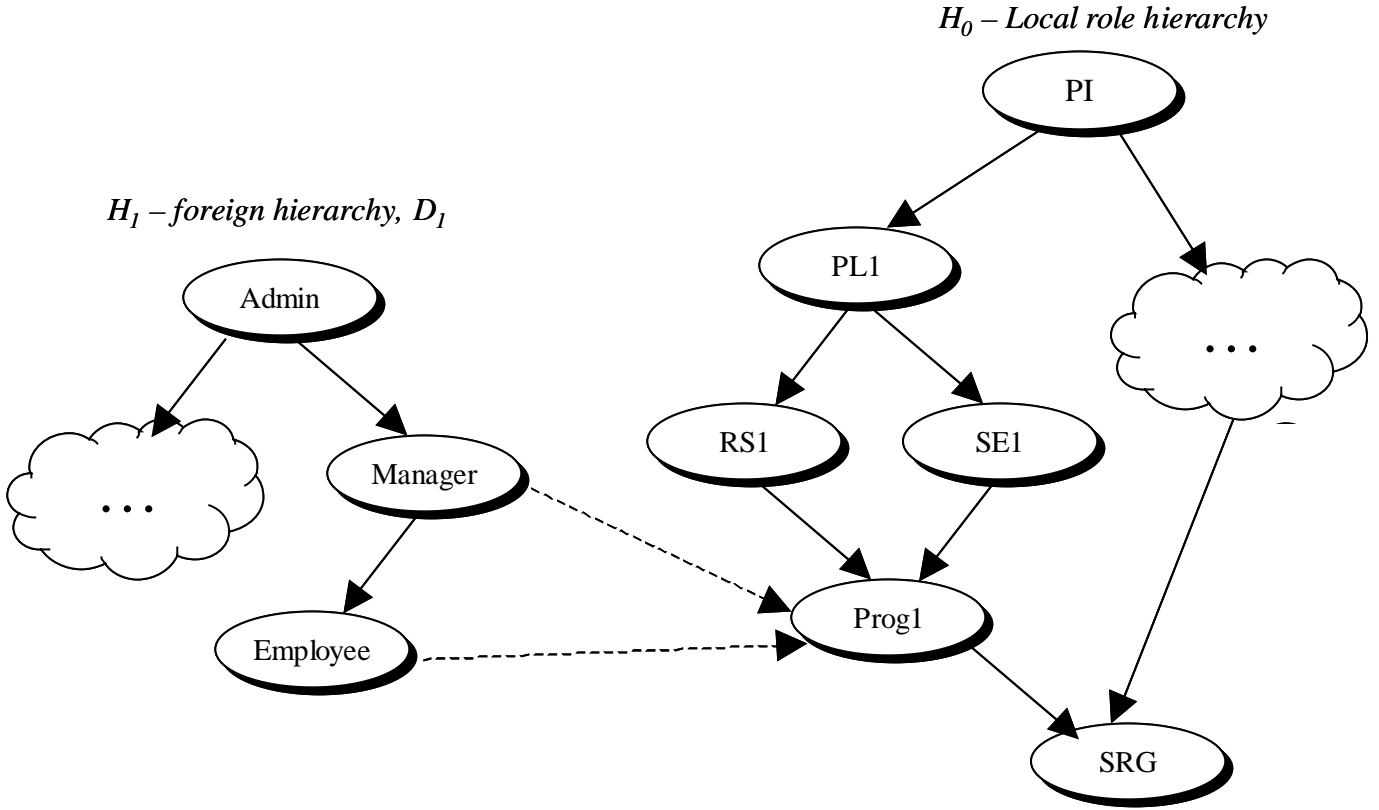


Figure 6: Sample mappings

3.4 Revoking Role Translations

This is also analogous to the way that ARBAC97 revoke users and permissions from roles [San99] and [San99b]. We define the following $can_revokeT$ relation for revoking inter-domain translations.

$$can_revokeT \subseteq IAR \times CR \times 2^{R_0}$$

We allow prerequisite conditions to be specified for revocations. Table 2 illustrates an example for the revoke model.

Table 2: $can_revokeT$ relation

IAR	Prerequisite condition	Authority range
SRGSO	$\neg mapped_to(Prog1) \wedge \neg mapped_to(Prog2)$	{SRG}
SO1	\emptyset	[Prog1, PL1]
SO2	\emptyset	[Prog2, PL2]
SSO	\emptyset	[SRG, PI] \cup {Guest}

From Table 2, the SRGSO can revoke foreign roles translations to SRG_{R_0} , as long as the foreign role cannot be translated to $Prog1_{R_0}$ or $Prog2_{R_0}$ (or to a local role senior to $Prog1_{R_0}$ or $Prog2_{R_0}$). SO1, SO2, and SSO have no restrictions on their revocation capabilities. Like ARBAC97 [San99] there is a notion of weak and strong revocations. Consider Figure 6. The role $Manager_{R_I}$ has an explicit translation to $Prog1_{R_0}$. If SO1 decides to *weakly revoke* this translation, then the edge from $Manager_{R_I}$ to $Prog1_{R_0}$ will be deleted. However, $Manager_{R_I}$ will still be able to translate to $Prog1_{R_0}$ since a role senior to it ($Employee_{R_I}$) has a non-transitive translation to $Prog1_{R_0}$.

A *strong revocation* is a series of weak revocations that eliminate indirect translations. In general, if we would like to revoke the translation from foreign role x_{R_I} to local role y , then we have to do the following series of weak revocations. Let r denote the role x_{R_I} .

- a) Weakly revoke every translation from remote role r to all local roles senior to y (including y itself.)

- b) For every foreign role in R_f that is junior to x_{R_f} and has a non-transitive translation to y (or a role senior to y) let r denote that foreign role and repeat step (a).

Note that if step (a) above goes beyond the revoking authority range of the officer who is strongly revoking a translation, the strong revocation would fail and no changes in the translations would take place. For example, if Manager $_{R_f}$ has a direct mapping to PI $_{R_o}$, then SO1's attempt to strongly revoke the translation from Manager $_{R_f}$ to Prog1 $_{R_o}$ will fail as PI $_{R_o}$ is outside SO1's authority range.

3.5 The Audit Component

Unfortunately, no system is 100% secure. In an environment with decentralized management, it is difficult to identify which security officer is responsible for a particular action. Hence, AIRBAC proposes the use of an audit facility. This facility permits the senior security officer to track changes in the system, discover security holes, detect intrusions and undo security breaches done by an intruder. The audit facility can be used for accountability purposes as well.

The audit facility should record changes done by security officers, especially the assignment and revocation of inter-domain translations, as well as foreign accesses to the local domain.

3.6 The Risk Assessment Component

Permitting roles in some foreign domain D_i to map to roles in the local domain implies that the local domain is investing some amount of trust in D_i . However, if D_i suffers from a security breach or starts misbehaving in some way, there should be some flexible and dynamic countermeasures that allow the local domain to shield itself from foreign security

breaches. Hence, we propose the *Risk Assessment Component* which can only be controlled by senior security officers. This component can be used to evaluate *at runtime* the likelihood that a threat will result in damage or compromise to some principal or component in the local domain. Threats are assessed in a number of ways, including conventional intrusion detection techniques. The senior security officer can manually adjust the risk parameter associated with a particular domain according to its recent "behavior" as captured by the audit facility. Once the risk value associated with a particular foreign domain goes over a certain threshold value, all role translations from that domain to the local domain are disabled.

4. Implementation

We are proposing a theoretical model. However, we have implemented a simple GUI tool (Figure 7) that can be used by users with interoperability administrative roles (IAR) to establish and revoke role translations. The tool is aware of the IAR hierarchy and attempts to authenticate a user to verify his administrative role. After a successful authentication, the regular local hierarchy is shown, and only local roles within the administrative authority of the user are enabled. The tool permits the user to load foreign hierarchies and create/revoke translations to enabled local roles only. Other aspects of the model, such as the Risk Assessment Component are still under development.

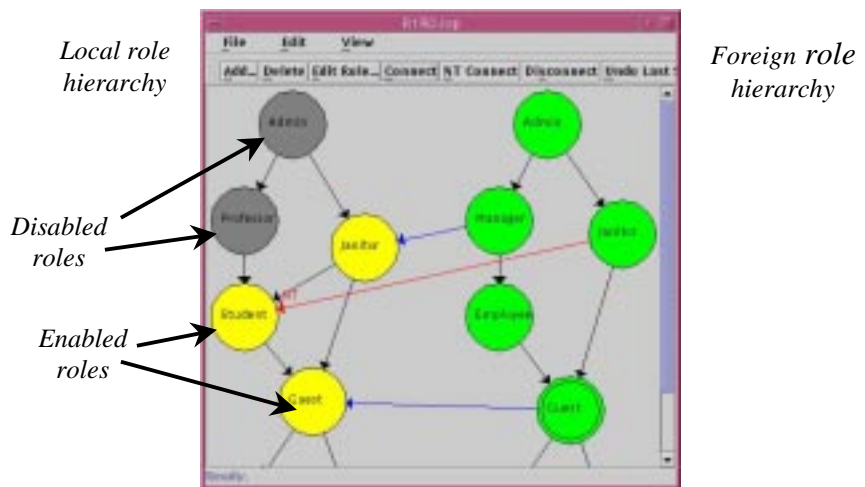


Figure 7: GUI Tool for managing translations

5. Future Work

We plan to complete the implementation of A-IRBAC 2000, particularly the Risk Assessment Component. This component has plenty of room for future extensions. For instance, foreign roles can be assigned individual risk values, capturing the fact that we may put more trust in particular roles within a foreign domain. Risk management could be managed through a more automated way with less intervention by the senior security officer. This includes the ability of the model to steer the risk values associated with foreign roles and domains dynamically, by assessing the probability that a security threat will result in compromise of some entities in the local domain. Threats can be automatically assessed by employing intrusion detection techniques, detailed auditing and pattern recognition. Automated risk management then guides the deployment of security measures and countermeasures to contain and control (or repair) the security compromise, as well as alerting the top security officers.

6. Conclusion

The A-IRBAC 2000 model extends IRBAC 2000 [Kap00] by employing RBAC to simplify the management of inter-domain role translations. The model utilizes concepts outlined in ARBAC97 [San99] and introduces new ideas that enable an organization to decentralize management, distribute authority between different administrative levels and keep track of responsibilities. Further, the model provides semantics for enforcing the global security policy with some ability to assess security threats on the fly and act accordingly.

References

- [Boe89] Boehm, B. *Tutorial: Software Risk Management*, IEEE Computer Society Press, Washington, D.C., (1989).
- [Cam00] R. Campbell, Z. Liu, D. Mickunas, P. Naldurg, and S. Yi. *Seraphism: Dynamic interoperable security ar-*

chitecture for active networks, IEEE OPENARCH 2000, Tel-Aviv, March 2000.

- [Kap00] Apu Kapadia, Jalal Al-Muhtadi, R. Campbell, D. Mickunas, *IRBAC 2000: Secure Interoperability Using Dynamic Role Translation*, The 1st International Conference on Internet Computing, June 26th - 29th, 2000, Monte Carlo Resort, Las Vegas, Nevada, USA.
- [Mal96] G. Malkin, *Internet Users' Glossary*, RFC 1983, network working group, August 1996.
- [San96] Ravi S. Sandhu, Edward J Coyne, Hal L. Feinstein and Charles E. Yocum. *Role-Based Access Control Models*. IEEE Computer, Volume 29, Number 2, Feb. 1996, pages 38-47.
- [San97] Ravi Sandhu, *Rationale for the RBAC96 family of access control models*, proceedings of the 1st ACM Workshop on Role-Based Access Control. ACM, 1997.
- [San99] Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer, *The ARBAC97 Model for Role-Based Administration of Roles*, ACM Transactions on Information and System Security, Vol. 2, No. 1, February 1999, pages 105-135.
- [San99b] Ravi Sandhu and Qamar Munawer, *The ARBAC99 Model for Administration of Roles*, 15th Annual Computer Security Applications Conference, December 6-10, 1999, Phoenix, Arizona.
- [Tin99] Qian, Tin, *Dynamic Authorization Support in Large Distributed Systems*, Ph.D. Thesis, University of Illinois at Urbana-Champaign, (1999).