A Model for Attribute-Based User-Role Assignment

Mohammad A. Al-Kahtani
George Mason University
malkahta@gmu.edu

Ravi Sandhu
SingleSignOn.net, Inc. &
George Mason University
sandhu@gmu.edu

Abstract

The Role-Based Access Control (RBAC) model is traditionally used to manually assign users to appropriate roles, based on a specific enterprise policy, thereby authorizing them to use the roles' permissions. In environments where the service-providing enterprise has a huge customer base this task becomes formidable. An appealing solution is to automatically assign users to roles. The central contribution of this paper is to describe a model to dynamically assign users to roles based on a finite set of rules defined by the enterprise. These rules take into consideration the attributes of users and any constraints set forth by the enterprise's security policy. The model also allows dynamic revocation of assigned roles based on conditions specified in the security policy. The model provides a language to express these rules and defines a mechanism to determine seniority among different rules. The paper also shows how to use the model to express Mandatory Access Controls (MAC).

1. Introduction

Role-Based Access Control (RBAC) has emerged as a proven and superior alternative to traditional discretionary and mandatory access controls [1, 2]. In RBAC permissions are associated with roles, and users are made members of appropriate roles, thereby acquiring the roles' permissions. This greatly simplifies management of permissions. Roles can be granted new permissions as new applications and systems are incorporated, and permissions can be revoked from roles as needed. Usually, the enterprise security officer(s) manually assign users to roles based on criteria specified by the enterprise.

Today, an increasing number of service-providing enterprises make their services available to their users via the Internet. RBAC can be used to manage users' access to the enterprise services and resources. In many environments, the number of users can be in the hundreds of thousands or millions. Typical examples are banks, utility companies, and popular Web sites, to name a few. This renders manual user-to-role assignment a formidable task.

An appealing solution is to automatically assign users to roles. This automatic assignment should be based on what attributes users have. It should also take into account any constraints laid down by the entity that provides the service.

In this paper, we describe a model to automatically assign users to roles based on a finite set of assignment rules defined by authorized people in the enterprise. These rules take into consideration the attributes users own and any constraints set forth by the enterprise. Users' attributes can be provided along with identification information or be retrieved from a database.

The model provides a language to express assignment rules and defines a mechanism to determine seniority among them. When certain conditions hold, the model also allows dynamic revocation of assigned roles. Since Mandatory Access Controls (MAC) are widely used in the military sphere, and are well understood, the paper also shows how to use the model to express MAC using the proposed language.

The paper is organized as follows. In section 2 we summarize related research. Section 3 describes our model. In sections 4 and 5 we show how our model can be used in two real life examples: one from the private sector and the other from the military sector. In section 6 we touch on issues that we have not explored in this paper, though they are closely related to the topic discussed. Section 7 concludes the paper.

2. Related Work

The central concept of RBAC is the role, which can be viewed as a semantic construct around which access control policy is formulated. Permissions are associated with roles. Users are assigned to appropriate roles based on factors such as their responsibilities and qualifications. Users can be easily reassigned roles. Roles can be granted new permissions, and organized in role hierarchies to reflect the organization's lines of responsibility and authority. In this work we adopted RBAC96 model presented in [2] after stripping out sessions, as shown in Figure 1.

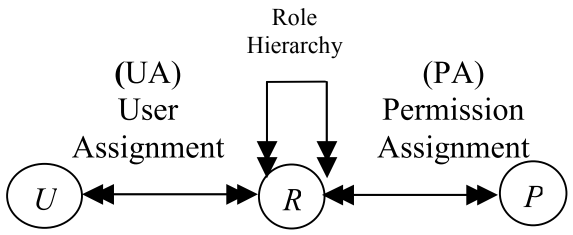

Figure 1 Simplified RBAC

When first developed, RBAC was intended for closed-enterprise systems in which the security administrator(s) assign roles manually to users. Park and Sandhu presented RBAC as a sound candidate to control users' access to resources and services in large-scale Web environments [3]. They identified architectures that can be used to implement RBAC on the Web. They also showed how existing technologies can be utilized to support these architectures. However, the architectures proposed were only in the context of enterprise-wide systems in which systems administrators assign users to roles on the basis of users' responsibilities in the enterprise.

In [4], Herzberg et al. presented a Trust Establishment (TE) system that defines the mapping of strangers to predefined business roles, based on certificates issued by third parties. Part of the proposed system is an XML-based Trust Policy Language to map users to roles using well-defined logical rules. Each role has one or more rules defining how a client can be assigned that role. The TE system gathers certificates related to a specific client and makes a decision regarding the client's eligibility for a specific role. The system proposed in [4] does not pay attention to relations that might exist among different rules. Another drawback in the TE system is that it is based on bottom-up buildup of the public key infrastructure (PKI), which imports all the issues related to PKI.

Another work that is closely related to [4] and to ours is found in [5] by Zhong, et al. They proposed a schema to use RBAC on the Web and a procedure for user-role assignment. Their schema assigns a client to a role based on legitimacy of information gathered, assignment policies, and the trustworthiness threshold specified by system administrators. Trustworthiness of a user is defined as clearance of the user. It represents the degree to which the enterprise believes that a user will not do harm to its Web site system. It is accumulated gradually over time and drops if harmful actions or potential harmful actions are discovered. There is a major drawback to this approach. A malicious user may logon to the system for long time without doing any suspicious acts. As time goes on, he acquires a high clearance, which might enable him to inflict damage on the system. Also, the scheme depends on many security parameters, which must be given initial values. It leaves determining these values to system administrator(s), but does not provide any guidelines on how to determine them.

Lightweight Directory Access Protocol (LDAP) is specifically targeted at management applications and browser applications that provide read/write interactive access to directories supporting the X.500 models [6]. Roles can be stored in directories and retrieved when needed. LDAP has been augmented to support dynamic groups. A dynamic group is an object with a membership list of distinguished names that is dynamically generated using LDAP search criteria. The dynamic membership list may then be interrogated by LDAP search and compare operations, and be used to identify a group's access control subjects [7]. This feature could be used to automatically assign users to roles in large enterprises. To retrieve the roles a specific client is entitled to assume, the filter in the search operation is configured to reflect the attributes the client has. When the search operation is executed, LDAP returns a list of the attributes extracted from each entry that matches the search filter specified in the search operation. The LDAP directory can be configured in such a way that returned attributes store the roles that match the search filter. However, implementing LDAP solely for the sake of dynamically assigning users to roles is an unwieldy solution. Also, LDAP returns a simple list of attributes (which represent roles in our case) with no logical structure attached to them. If, for example, a client can assume one of two mutually exclusive roles, LDAP does not provide a simple mechanism to express this.

Yao et al. [8] present an RBAC model that does not recognize role hierarchies explicitly. Instead, they propose a role activation dependency that is dynamic. A set of parameterized rules governs the activation of every role. Their model is rich in terms of expressing the rules, and associated conditions. However, we think that eliminating role hierarchies is a debatable issue to say the least. Role hierarchies have values not only from the user-assignment perspective of roles but also from the permission-assignment perspective. Also, by making the hierarchies implicit via side effects of role activation rules, the model does not explicitly capture various relations that might exist among roles.

3. The Model

In the model we propose, we modify RBAC such that it becomes rule-based, as discussed below. Thus, we named it Rule-Based RBAC or RB-RBAC. In this model, an enterprise defines the set of rules that are triggered to automatically assign users to roles. These rules take into account:

- The attributes of the client that are expressed using attributes' expressions as defined by the language provided by the model.
- Any constraints on using roles.

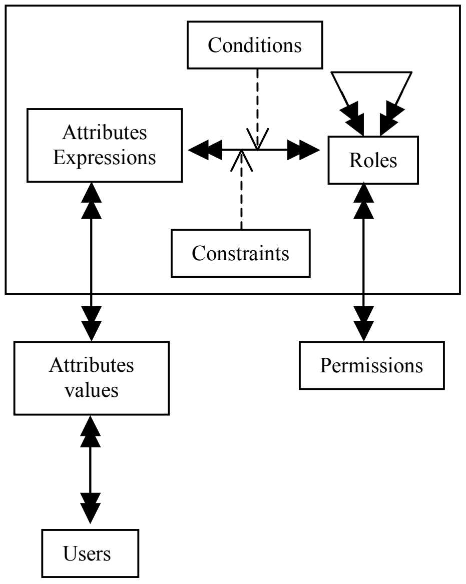

Figure 2 RB-RBAC model

Figure 2 shows that users have many-to-many explicit relation with attribute values. Further, they have many-to-many implicit relation with attribute expressions. One user could have one or more attribute expressions depending on the information he provides. Conversely, two or more users may provide identical attribute expressions. A specific attribute expression corresponds to one or more roles. An example of a rule that yields multiple roles is when a client is entitled to several mutually exclusive roles. The figure also shows that a role may be hierarchically related to one or more roles (in the usual partial order of roles). The figure also shows that a role may correspond to one or more attribute expressions.

In order to assign the role(s) specified by the rule to a client, the following must hold:

- The client must provide attributes that satisfy the attribute expressions.
- All constraints must be observed.

Conditions allow dynamic revocation of role assignment if a condition required by the assignment rule can no longer be satisfied.

3.1. Assumptions

1. Users are properly authenticated before our model is triggered to assign them roles.
2. Role-permission assignment is relatively stable compared to user-role(s) assignment. Thus automated permission-to-role assignment constitutes a good candidate for future work.
3. Users' attributes are provided along with the authentication information or can be fetched from databases.
4. The number of users is much larger than the number of roles (such as, hundred of thousands or millions of users versus less than a hundred roles).

3.2. The Language

Using context-free grammar, we define the language given in Figure 3, which is largely self-explanatory.

3.3.1. The terminal symbols: {AND, OR, XOR, NOT, <, =, >, ≤, ≠, ≥, IN, "SUBJECTED TO", "REVOKED IF NOT", 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

3.3.2. The non-terminal symbols: {Attribute_Expression, Attribute_Pair, Relation_Operator, Operator, Attribute, Roles, Constraints, Num, Digit, Conditions,Set,Range,Role,Attribute_Value}
The values of the non-terminal symbols Set, Attribute, Attribute_Value, Role and Constraint are specified by the organization.

3.3.3. The Start symbol: Rule

3.3.4. The production rules (in BNF notation):

Rule ::= Attribute_Expression **SUBJECTED TO** [Constraints] **REVOKED IF NOT** [Conditions]→ Roles.
Attribute_Expression ::= Expression
Conditions ::= Expression
Expression ::= Attribute_Pair
  | Expression Operator Expression
  | (Expression Operator Expression)
Attribute_Pair ::= Attribute Relation_Operator Attribute_Value
  | Attribute [NOT] IN Set
  | Attribute [NOT] IN Range
Roles ::= [NOT] Roles
  | Roles Operator Roles
  | (Roles Operator Roles)
  | Role
Constraints ::= Constraint
  | Constraint Operator Constraint
  | (Constraint Operator Constraint)
Operator ::= AND | OR | XOR
Relation_Operator ::= < | = | > | ≤ | ≠ | ≥
Attribute ::= {*specified by organization*}
Attribute_Value ::= {*specified by organization*}
Set ::= {*specified by organization*}
Range ::= (Num..Num)
Num ::= Num Digit
Digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
Role ::= {*specified by organization*}
Constraint ::= {*specified by organization*}

Figure 3: RB-RBAC Language

We kept the language simple but extendable to show the usability of the model. We are working on many possible extensions, some of which are discussed later in this paper.

3.3. The Seniority Levels

The attributes provided by a user may not literally meet the Attribute_Expression requirement of a specific rule. In some cases, they might provide more than what is called for by the rule. In real life, such a user is entitled to assume the role specified by that rule. The language as defined above does not provide a mechanism to compare the attributes provided by the users to those required by a rule. Also, it might be desirable to compare two rules. To do this, we introduce seniority levels:

- Attributes' values specified by the organization are given seniority levels showing what value dominates what. In case of numeric values, seniority automatically follows the normal order of values when we have the following syntax:
  Attribute ≥ numeric value or
  Attribute > numeric value
  However, seniority levels go in reverse order with numeric values when the syntax is in the following form:
  Attribute < numeric value or
  Attribute ≤ numeric value
  In case of equality, inequality, sets, and ranges, seniority levels must be manually specified.
- Two (Attribute_Expression)s are said to be *comparable* only if
  1. They have identical structures (syntax) according to our notation, and
  2. They are subjected to the same constraints
- Only comparable (Attribute_Expression)s are tested for seniority.
- We use the symbol ≥ to denote dominance.
- We say Attribute_Expression$_i$ dominates (≥) Attribute_Expression$_j$:
  If ∀ x,y: x is the ith attribute value ∈ Attribute_Expression$_i$ and y is the ith attribute value ∈ Attribute_Expression$_j$,
  then
  Seniority Level of (x) ≥ Seniority Level of (y)
- A user with Attribute_Expression$_i$ is entitled to roles specified by rules whose (Attribute_Expression)s are dominated by Attribute_Expression$_i$.
- Rule$_i$ (with Attribute_Expression$_i$) is said to be senior to Rule$_j$ (with Attribute_Expression$_j$) only if Attribute_Expression$_i$ ≥ Attribute_Expression$_j$.
- A senior rule inherits all the roles produced by any of its junior rules. This approach, though intuitive, introduces two issues, which we describe here informally:

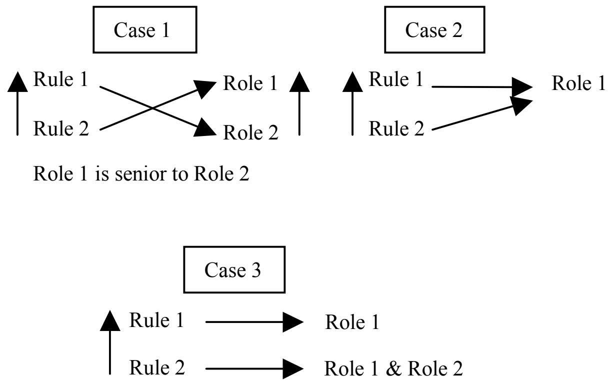

Figure 4 Examples for Redundancy Among Rules

1. Redundancy: This occurs when a senior rule yields a role or group of roles, which are yielded by a junior rule. Figure 4 shows examples of redundant rules.
   In the figure, the vertical arrows indicate the direction of seniority. In case 1, for example, Rule 1, which is senior to Rule 2, produces Role 2. However, Rule 2 produces Role 1, which is senior to Role 2. In other words, a user whose attributes satisfy Rule 1 can obtain Role 1 either directly via Rule 2 or indirectly by invoking Rule 1, which inherits it from Rule 2. Deleting a senior rule eliminates redundancy without diminishing the roles, and hence the privileges a client is supposed to obtain. The same solution applies to cases 2 and 3.
2. Inconsistency: Several scenarios could render a set of rules inconsistent, as illustrated in Figure 5. One reason for inconsistency is the mutually exclusive roles. We distinguish between 2 types of these roles:
   a. Mutual exclusion that must be observed throughout all the assignment rules or among rules that have no seniority relationship among them. Enforcing mutual exclusion of roles in this case requires using constraints. Assume that the security policy of the enterprise considers Role 1 and Role 2 mutually exclusive roles. A user may try to activate them simultaneously by providing attributes that satisfy Rule 1 and, immediately after that, providing attributes that satisfy Rule 2. To prevent this, the rules in case 1, shown in the figure, could be expressed as follows:
   - Attribute Expression for Rule 1 SUBJECT TO user not currently enrolled in Role 2 $\rightarrow$ Role 1
   - Attribute Expression for Rule 2 SUBJECT TO user not currently enrolled in Role 1 $\rightarrow$ Role 2

In case 2, Rule 2 explicitly violates the security policy so to remove the inconsistency we delete the senior rule and rewrite the junior rule as follows:

Rule 2 → Role 1 XOR Role 2.

b. Mutual exclusion that must be observed throughout assignment rules that are related via seniority. The inconsistency in case 3 was introduced via an explicit violation of the security policy and can be eliminated in a way similar to the one used in case 2. In case 4, a user who successfully triggers Rule 1 will be directly forbidden from assuming Role 1, but he can assume that role indirectly via inheriting Rule 2, which is junior to Rule 1. There are 2 approaches to deal with this inconsistency:

- Eliminating the junior rule, which results in fewer permissions available to users.
- Eliminating the senior rule and thus retaining the current sum of permissions.

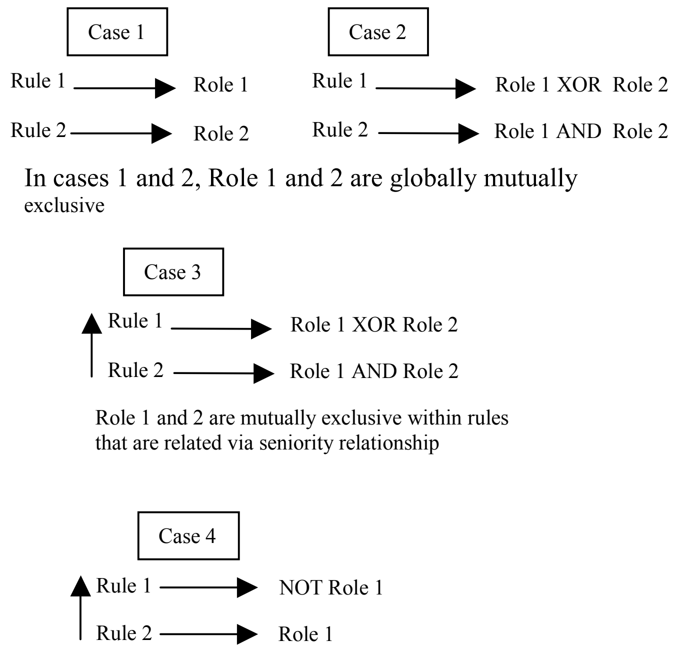

Figure 5 Examples for Inconsistency

4. Case I: Online Entertainment Store

An online entertainment store provides movies, games, documentary films, etc. The content of the material provided is rated according to a hypothetical rating system as shown in Table 1. Using RBAC terminology, levels correspond to roles, which, in turn, correspond to permissions. Roles are totally ordered in this example. When users logon, the attributes they provide determine the highest level they can obtain. For the sake of the discussion, we will consider 2 attributes: the age of the users and the country from which they initiate the service request.

Table 1 New Visual Material Rating System

Rating Level	Content of Material Displayed: "Permissions"	Corresponding Role
L1: Strict	• No violation of ethics • No foul language • No sexual language / scenes • No scary scenes	Child
L2: Less Strict	• Normal social behavioral patterns • No foul language • No sexual language /scenes • No scary scenes	Juvenile
L3: Liberal	• Normal social behavioral patterns • Moderate foul language • Moderate sexual language/scenes • Moderate scary scenes	Adolescent
L4: Graphic	• Extreme social behavioral patterns • Foul language • Explicit sexual language/scenes • Scary scenes	Adult

4.1. Attributes' Representation

4.1.1. Age

Table 2 shows how users who belong to different age groups are assigned to roles described in Table 1.Using the language, the Web site administration can specify the following non-terminal items:

i) Attribute ::= age
ii) Attribute_Value ::= 3, 11, 16, 18
iii) Role ::= Child | Juvenile| Adolescent| Adult

Since no constraints or conditions were specified, the following rules are produced:

- Rule 1:: (Age ≥ 3) → Child
- Rule 2:: (Age ≥ 11) → Juvenile
- Rule 3:: (Age ≥ 16) → Adolescent
- Rule 4:: (Age ≥ 18) → Adult

4.1.2. Country

Different countries have various laws regarding the access they permit their citizens to visual material.

Countries including China, India, Saudi Arabia, Egypt and Singapore do not allow materials that have explicit sexual content. Also, some countries outlaw materials containing symbols that represent certain ideologies, religious values, etc. Failing to abide by these laws may subject the enterprise to litigation. In November 2000, a French court ordered Yahoo! to devise a way to block Nazi paraphernalia from being auctioned through its site in France. The court also said Yahoo! would be charged a fine equivalent to $13,905 each day for supporting the Nazi items on its auction site [9].

Table 2 Attribute-Role Table for Attribute "Age"

Age	Role
≥ 3	Child
≥11	Juvenile
≥16	Adolescent
≥18	Adult

Based on our rating system defined above, we construct Table 3, which shows a hypothetical situation linking countries to roles. Set {A..Z} contains all countries of the world.

Table 3 Attribute-Role Table for Attribute "Country"

Country	Role
Country in {A..Z}	Juvenile
Country in {{A..Z} – {Saudi, Sudan}}	Adolescent
Country in {{A..Z} – {China, India, Saudi, Sudan, Egypt, Indonesia, Malaysia, Singapore}}	Adult

Note that if a country is in {A..Z}, then by RBAC definition, users in that country can assume the role Child since it is junior to Juvenile. Using the language, the Web site administration can specify the following non-terminal items:

i) Attribute ::= country
ii) Set ::= {A..Z}
| {{A..Z} – {Saudi, Sudan}}
| {{A..Z} – {China, India, Saudi, Sudan, Egypt, Indonesia, Malaysia, Singapore}}
iii) Role ::= Juvenile | Adolescent | Adult

Based on the above, the following rules are produced:

- Rule 1:: (Country IN {A..Z}) → Juvenile
- Rule 2:: (Country IN {{A..Z} – {Saudi, Sudan}}) → Adolescent
- Rule 3:: (Country IN {{A..Z} – {China, India, Saudi, Sudan, Egypt, Indonesia, Malaysia, Singapore}}) → Adult

Assuming that the security policy of the online store calls for considering age and geographical location simultaneously, then we can use the language provided by the model to specify the following rules:

- Rule 1:: (Age ≥ 3) AND (country IN {A..Z}) → Child
- Rule 2:: (Age ≥ 11) AND (country IN {A..Z}) → Juvenile
- Rule 3:: (Age ≥ 16) AND (country IN {{A..Z} – {Saudi, Sudan}}) → Adolescent
- Rule 4:: (Age ≥ 18) AND (Country IN {{A..Z} – {China, India, Saudi, Sudan, Egypt, Indonesia, Malaysia, Singapore }}) → Adult

4.2. Implementing Seniority Levels

Assume that the security officer in the entertainment store assigned seniority levels to attribute values as in Table 4. Based on the above, if a client owns attribute expression (Age ≥ 16) AND (country IN {A..Z}), then Rule 2 is triggered and the client gets role Juvenile because:

- Age value (16) dominates Age value (11)
- The client's attribute "country" has a value that is identical to the one required by Rule 2

Table 4 Attributes and Seniority Levels

Attribute Value	Seniority Level
Attribute = Age	Follows the regular order of numeric values: a value of 16 is senior to 3.
Attribute = Country	
Country in {A..Z}	1
Country in {{A..Z} – {Saudi, Sudan}}	2
Country in {{A..Z} – {China, India, Saudi, Sudan, Egypt, Indonesia, Malaysia, Singapore}}	3

4.3. Implementing MAC Using the Language

The language we introduced above can be used to implement MAC. In the visual entertainment store case, we can assume having a security lattice similar to the one in Figure 6. In our example, (Simple Security Property) is enforced so that a client ("subject" in MAC terminology) can view (read) materials (objects) that have security labels that are not superior to his clearance. The client can also make comments or post queries about materials provided by the store. A client at Adult level, for example, can write comments about materials at his level. If he wants to write about materials at Child level, he has to login as a subject at Child level. A client at Child level can not write a comment that at Adult level and hence, (Strict *-Property) is enforced.

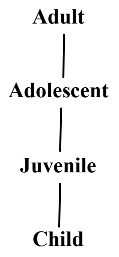

Figure 6 Security Lattice

Osborn et al showed in [10] that the lattice in Figure 6 can be converted to two role hierarchies as shown in Figure 7. In this case, we have the following construction:

- Roles = {Adult Read, Adolescent Read, Juvenile Read, Child Read, Adult Write, Adolescent Write, Juvenile Write, Child Write}.

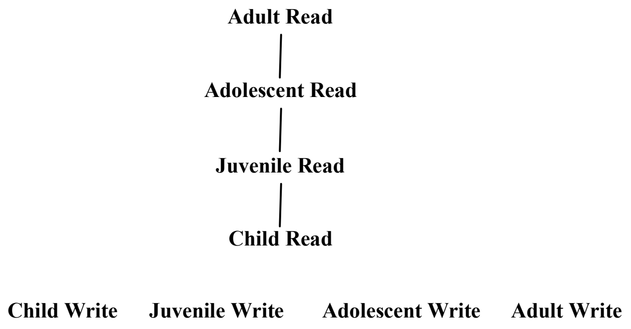

Figure 7 Role Hierarchies for the Lattice in Figure 6

- Role hierarchy consists of two disjointed parts. The first is a role hierarchy for the "read" roles: {Adult Read, Adolescent Read, Juvenile Read, Child Read}. This hierarchy has the same partial order as dominance relation (≥ as in MAC). The second part is composed of incomparable "write" roles: {Adult Write, Adolescent Write, Juvenile Write, Child Write}.
- Constraint on User assignment: Each user is assigned to exactly two roles: xR and xW where x is the label assigned to the user.

Using the language we defined, we can specify:

Role ::= Adult Read | Adolescent Read | Juvenile Read | Child Read | Adult Write | Adolescent Write | Juvenile Write | Child Write

Each read role has a companion write role and a user must be assigned to both roles simultaneously in order for the user to function properly. Table 5 shows the correspondence among read and write roles. Using the language and the correspondence shown in Table 5, we write the following:

Rule 1:: (Age ≥ 3) AND (country IN {A..Z})
→ CR AND CW

Rule 2:: (Age ≥ 11) AND (country IN {A..Z})
→ JR AND JW

Rule 3:: (Age ≥ 16) AND (country IN {{A..Z} – {Saudi, Sudan}})
→ DR AND DW

Rule 4:: (Age ≥ 18) AND (Country IN {{A..Z} –{China, India, Saudi, Sudan, Egypt, Indonesia, Malaysia, Singapore}})
→ AR AND AW

Table 5 Companion Read and Write Roles

Read Roles	Write Roles
Adult Read (AR)	Adult Write (AW)
Adolescent Read (DR)	Adolescent Write (DW)
Juvenile Read (JR)	Juvenile Write (JW)
Child Read (CR)	Child Write (CW)

5. Case II: Military Equipment Spare Parts

A corps maintains supplies of military equipment spare parts that are usually stored at different geographical locations. Maintenance of military equipment consists of multiple echelons, with varying degrees of depth from organizational (OM), to intermediate (IM), to depot-level maintenance (DM) [11]. Maintenance personnel at different levels need access to logistical databases to order spare parts they need, locate the nearest warehouse of a needed spare part, or follow up the status of spare parts and repair orders. For security reasons, the information displayed to maintenance personnel and the privileges they exercise depend on a group of factors which includes:

a) Unit's geographical location
b) Type of equipment
c) Unit's alert status
d) Maintenance Level: usually, OM, IM, and DM are performed by different maintenance entities. For simplicity, we will ignore the IM level.

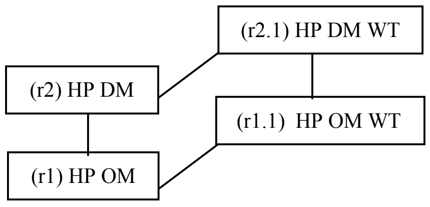

Figure 8 Roles' Maintenance Levels and Alert Status

In our discussion, we will consider the case of maintaining the High Powered Tracking Radar (HP) at two maintenance levels during two alert statuses: peacetime and wartime (WT). Variations in alert status or maintenance level result in hierarchical role structure shown Figure 8. The roles and permissions for this case are illustrated in Table 6.

Table 6 Roles and Permissions for HP

Role	Permissions
r1: HP OM	• request parts from local warehouse • follow up orders previously made • inquire in local database
r2: HP DM	• request parts from all warehouses • follow up orders previously made • inquire in national databases
r1.1: HP OM WT	• r1 + • give priority to orders
r2.1: HP DM WT	• r2 + • give priority to orders

5.1. Attributes' Representation

We will represent Maintenance Level and Alert Status. The organization can specify the following non-terminal items:

i) Attribute ::= Maintenance Level | Alert Status
ii) Attribute_Value ::= OM | DM | Peacetime | Wartime
iii) Role::= r1 | r2 | r1.1 | r2.1

Based on the above, we can use the language to define the following rules:

Rule 1:: (Maintenance Level = OM AND Alert Status = Peacetime) → r1
Rule 2:: (Maintenance Level = DM AND Alert Status = Peacetime) → r2
Rule 3:: (Maintenance Level = OM AND Alert Status = Wartime) → r1.1
Rule 4:: (Maintenance Level = DM AND Alert Status = Wartime) → r2.1

5.2. Implementing MAC Using the Language

MAC is widely used in military organizations so it is reasonable to investigate the possibility of implementing it using the language defined above. We modify our example by introducing a second equipment type: Missile. Alert status is ignored to keep this example simple. This yields the hierarchical classes and compartments shown in Figure 9, which produces the lattice structure shown in the same figure.

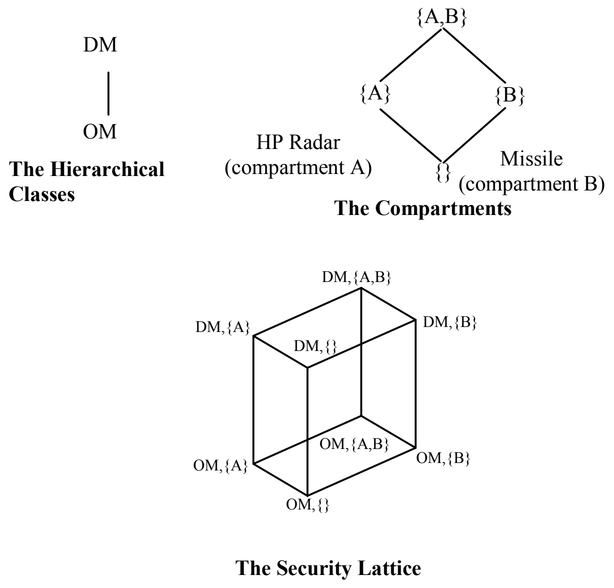

Figure 9 Hierarchical Classes and Compartments

Using the method expounded upon in [10], we can transform the lattice into RBAC role hierarchies as shown in Figure 10. Roles (DM, {A,B}-R), (DM,{A,B}-W), (OM-R) and (OM-W) were introduced by the lattice but they have no equivalence in our example.

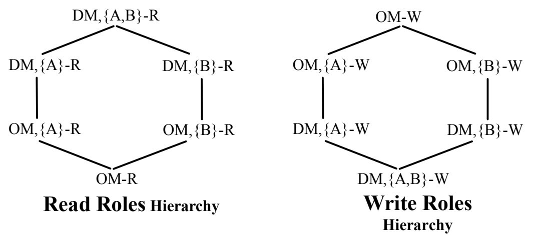

Figure 10 RBAC Role Hierarchies

We can express the security lattice in terms of an RBAC model using the following construct:

- Roles are shown in Figure 10.
- We have two disjointed Role hierarchies:

1. A "read" role hierarchy that has the same partial order as dominance relation ($\geq_{MAC}$). Role DM,{A}-R allows a user to read HP Radar part information found in any database nationwide, while OM,{A}-R allows reading that data from the local database.
2. A "write" role hierarchy that has a partial order that is the inverse of dominances ($\geq_{MAC}$). Role OM-W dominates all other write roles, which enables users to write data to levels in the security lattice that are higher than their own.

- Constraint on User assignment: Each user is assigned to exactly two roles: xR and OM-W, where x is the label assigned to the user. OM-W is the write role corresponding to the lowermost security level according to ($\geq_{LBAC}$).

Each user is assigned a read role yR and its companion write role yW. Table 7 shows the roles with hypothetical permissions.

Table 7 RBAC Roles and Permissions

RBAC Roles	Permissions
OM,{A}-R	• inquire in local database about HP parts
DM,{A}-R	• inquire in all databases about HP parts
OM,{A}-W	• write report to HP OM and DM levels
DM,{A}-W	• write report to HP DM level only
OM,{B}-R	• inquire in local database about missile parts
DM,{B}-R	• inquire in all databases about missile parts
OM,{B}-W	• write report to missile OM and DM levels
DM,{B}-W	• write report to missile DM level only

Read and write roles are assigned according to (Liberal *-Property) as shown in Table 8. Each line in the last column of the table shows the roles that are assigned to the user at any one time.
The organization can specify the following non-terminal items:

i) Attribute ::= Maintenance Level
| Equipment Type
ii) Attribute_Value ::= OM | DM | HP | Missile
iii)Roles: {*read and write roles in Table 8}*

We can use the language to specify the following rules:

Rule 1:: (Maintenance Level = OM AND
Equipment Type = HP)
→ (OM,{A}-R AND OM,{A}-W)
Rule 2:: (Maintenance Level = DM AND
Equipment Type = HP)
→ ((DM,{A}-R AND DM,{A}-W))
XOR
((OM,{A}-R AND OM,{A}-W))
Rule 3:: (Maintenance Level = OM AND
Equipment Type = Missile)
→ (OM,{B}-R AND OM,{B}-W)
Rule 4:: (Maintenance Level = DM AND
Equipment Type = Missile)
→ ((DM,{B}-R AND DM,{B}-W))
XOR
((OM,{B}-R AND OM,{B}-W))

The language allows us to capture mutually exclusive roles as shown in the rules above.

Table 8 Lattice Labels and Corresponding RBAC Roles

Label	RBAC Roles		Roles Assigned to a User
	Read Roles	Write Roles	
OM,{A}	OM,{A}-R	OM,W	(OM,{A}-R AND OM,{A}-W)
DM,{A}	DM,{A}-R	OM,W	(DM,{A}-R AND DM,{A}-W) XOR (OM,{A}-R AND OM,{A}-W)
OM,{B}	OM,{B}-R	OM,W	(OM,{B}-R AND OM,{B}-W)
DM,{B}	DM,{B}-R	OM,W	(DM,{B}-R AND DM,{B}-W) XOR (OM,{B}-R AND OM,{B}-W)

6. Discussion

The language we present is simple and can be used to express situations beyond the study cases provided in this paper. For example, a user's membership in a role is an attribute of paramount importance in many applications. Assume that the security policy of an enterprise allows a client to be assigned a "team leader" role only if he is already a member in a "programmer" role. This can be represented as follows:

Attribute_Pair ::= Attribute IN Set

where

Attribute ::= required-role {*attribute's name is specified by the enterprise*}
Set ::= {programmer}

Similarly, suppose that the security policy allows the client to retain the role that is currently assigned to him during regular working hours only. This condition can be captured as:

Attribute_Pair ::= Attribute IN Range

where

Attribute ::= time {*defined by the organization*}
Range ::= 900..1700 {*from 9 am to 5 pm*}

When time is not within this range, the role is automatically revoked from the client.
However, the language needs to be extended to allow more expressive power. One area that we are working on is devising a way for expressing constraints. Also, the interaction between seniority among rules and seniority among roles is yet to be explored. For example, in many scenarios, by examining seniority among rules, one can derive a hierarchy among roles. By going back to the rules derived from Table 2, one can deduce that role Child is junior to Juvenile.
Our model uses seniority levels to compare a client's attributes to existing assignment rules, or to compare two assignment rules. We will work to make seniority level comparisons less restricted to allow insight about relationships among rules. We will also explore the usability of our model in role-to-permission assignment. More importantly, we will extend our

model to allow cross-domain dynamic user-to-role assignment to reduce administrative work for security administrators.

7. Conclusion

We have described a model to dynamically assign users to roles based on a finite set of rules defined by authorized people in an enterprise. We believe that our model will be useful in automatically—rather than manually—managing users to role(s) assignment and revocation in enterprises with large client bases.

8. References

[1] R. Sandhu, V. Bhamidipati, and Q. Munawer, "The ARBAC97 Model for Role-based Administration of Roles". *ACM Transactions on Information and System Security.* Vol.2, No.1, Feb. 1999, pages 105-135.

[2] R. Sandhu, E. Coyne, H. Feinstein and C. Youman, "Role-based access control model", *IEEE Computer*, 29(2), Feb. 1996.

[3] J. Park, R. Sandhu and G. Ahn, "Role-based Access Control on the Web", *ACM Transactions on Information and System Security,* Vol. 4, No 1, 2001.

[4] A. Herzberg, Y. Mass, and J. Mihaeli, "Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers", *Proc of the 2000 IEEE Symposium on Security and Privacy,* 2000.

[5] Y. Zhong, B. Bhargava, and M. Mahoui, "Trustworthiness Based Authorization on WWW", In *IEEE workshop on "Security in Distributed Data Warehousing",* New Orleans, Oct. 2001.

[6] Lightweight Directory Access Protocol (v3), RFC2251, December 1997.

[7] Dynamic Groups for LDAPV3 draft-haripriya-dynamicgroup-00.txt, October 2001.

[8] W. Yao, K. Moody, J. Bacon, "A Model of OASIS Role-Based Access Control and its Support for Active Security", *SACMAT'01,* Chantilly, Virginia, USA, May 3-4, 2001.

[9]http://www.guardian.co.uk/international/story/0,3604,400649,00.html

[10] S. Osborn, R. Sandhu, and 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, May 2000, pages 85-106.

[11] http://www.defenselink.mil/ra/rfpb/chapter_5.html