Risk Analysis in Access Control Systems

J. Ma†, K. Adi†, M. Mejri‡, L. Logrippo†

† *Department of Computer Science and Engineering*
Université du Québec en Outaouais
Québec, Canada.
‡ *Computer Science and Software Engineering Department,*
Laval University, Québec, Canada
Email: {ji.ma,kamel.adi,luigi}@uqo.ca, mejri@ift.ulaval.ca

***Abstract*—Commonly known access control systems respond to users' requests to perform actions on protected objects by giving binary answers such as $permit$ or $deny$. The decisions are taken on the basis of access control policies, where the risk of allowing access is not necessarily taken into explicit consideration. In this paper, we introduce RBAC^R model (Role Based Access Control Model with Risk), in which each access control decision is taken after consideration of risk assessment. The proposed risk assessment method considers partial orderings on objects and actions to capture the notions of importance of objects and criticality of actions, and determines the risk of assigning a specific role to a specific user. The case of role delegation is also considered.**

Keywords: Access control, risk analysis, RBAC, RBAC^R, model.

I. Introduction

Computer based access controls prescribe not only which users are allowed to access a specific object, but also which types of access are permitted. Within a role based access control system, access decisions, i.e., granting or denying permissions are taken in consideration of the roles of individual users. For example, a user with role "doctor" normally has different access rights than a user with role "nurse".

In this paper, we investigate an extension of the basic role based access control (RBAC) model [2], [4], [5], [6], called RBAC^R model (Role Based Access Control Model with Risk) that includes concepts useful for risk evaluation. Risk analysis in access control systems involves many factors. In role based access control systems, users hold certain roles, and may or may not be allowed to access the objects requested and take actions on these objects. For any access request, apart from identifying the user and its role, the access control system must know: which object is requested and what action may be applied by the user on the object.

We add a risk parameter in RBAC rules in order to be able to evaluate risk values associated with access requests. Access permission is defined as a pair of an action and an object; a role is then defined as a set of permissions. We introduce an ordering relation on sets of objects and actions to capture the notions of importance of objects and criticality of actions. From these posets, we automatically build an ordering relation on permission sets (corresponding to roles). The risks for assigning users to roles, which is a critical issue in RBAC [10], [1], [11], is then evaluated. We also investigate the risk for delegation.

Most research on risk analysis in access control systems is based on binary values (meaning: risk present or absent, access allowed or denied) or manually-defined values. One of the earliest proposals for access control was made by Bell and LaPadula [3] in 1976. The Bell-LaPadula model (BLM) is a binary model, it specifies controls for reading files that are necessary to preserve various degrees of data confidentiality, and also controls for writing to ensure that data is not copied in a container where appropriate confidentiality is not guaranteed. Decision making in this model does not depend on risk analysis. For a subject with unclassified clearance, access to secret or top secret files is equally forbidden, whereas a risk model might consider the second access more risky than the first.

Aziz *et al.* [2] discuss reconfiguring role based access control policies using risk semantics. Dimmock *et al.* [4] discuss trust and risk analysis in role-

based access control policies. Kondo *et al.* [5] discuss extending the RBAC model for large enterprises and quantitative risk evaluation. Nissanke and Khayat [6] discuss risk based security analysis of permissions in RBAC systems.

In this paper, we propose a computable model of RBAC with the notion of risk, in which a formal risk related semantics is defined. Our method automatically computes risks associated with access requests. It deals with direct as well as delegated permissions. We show that the risk of granting permission can be evaluated at the moment of the decision, using risk parameters and risk levels known at the moment.

The rest of this paper is organized as follows. Section 2 introduces role based access control systems. Section 3 introduces the RBAC^R model (role based access control model with Risk) for risk analysis. Section 4 discusses how to reason about access control systems with risk assessment. Section 5 is a brief discussion about implementation issues. Section 6 concludes this paper and discusses further works.

II. Role Based Access Control Systems

Definition 1 (Role Based Access Control System)**.** *A role based access control (RBAC) system, denoted by M_{rbac}, is a 6-tuple,*

$$M_{rbac} = \langle \mathcal{U}, \mathcal{R}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{AR} \rangle,$$

where

- *$\mathcal{U}$: a set of subjects,*
- *$\mathcal{R}$: a set of roles,*
- *$\mathcal{O}$: a set of objects,*
- *$\mathcal{A}$: a set of actions,*
- *$\mathcal{P}$: a set of permissions, $\mathcal{P} \subseteq \mathcal{A} \times \mathcal{O}$, and*
- *$\mathcal{AR}$: a set of assignment relations.*

The set of assignment relations, $\mathcal{AR}$, includes the following relations:

- RA: role assignment relation, $RA \subseteq \mathcal{U} \times \mathcal{R}$, and
- PA: permission assignment relation, $PA \subseteq \mathcal{R} \times \mathcal{P}$.

A user may hold zero or more roles, and a role may possess one or more permissions.

Generally, for security considerations the system may not trust anyone, but it trusts:

- those facts that come from system configuration (role assignment, permission assignment, etc), and
- access control policies, which are precisely specified and verified.

Therefore, if we formalize these facts and polices to establish a theory (axioms and inductive rules) for a given RBAC system, then the system can reason with this theory for decision making. In this paper, we focus on the risk analysis and reasoning techniques that can be used in RBAC systems.

III. RBAC^R Model

In this section, we introduce RBAC^R model (Role Based Access Control Model with Risk), where risk assessment is considered and applied to evaluate risk values associated with access requests in a given RBAC system.

In RBAC systems, there may be a number of risk analysis functions, which are used to calculate risk values related to recognized threats. Risk analysis functions should play an important role in reasoning and decision making in such systems. Adding the risk assessment parameter to RBAC systems, we get the RBAC^R model.

Definition 2 (RBAC^R model)**.** *Let $M_{rbac} = \langle \mathcal{U}, \mathcal{R}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{AR} \rangle$ be a role based access control system. Then:*

$$M_{rbac^r} = \langle \mathcal{U}, \mathcal{R}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{AR}, \mathcal{RF} \rangle$$

is called the Role-Based Access Control model with Risk corresponding to M_{rbac}, where $\mathcal{RF}$ is a set of risk analysis functions.

In the RBAC^R model, decision making depends on risk assessment. The risk functions set, $\mathcal{RF}$, is a major component in the RBAC^R model and makes this model different from the classical RBAC model. To define the set of risk analysis functions, we need to identify those situations or treats that may lead to risk, and then find appropriate functions for calculating the risk values related to the recognized treats. In the following, we discuss how to define appropriate risk analysis functions for two main ingredients in RBAC: role assignment and delegation.

Role assignment is a major component of an RBAC system and risk analysis is a main consideration of role assignment. In an RBAC system, users hold certain roles. Whether a user is allowed to conduct an action on a required resource depends on the role that the user holds. Therefore, in an RBAC system, it is important to determine the risk of assigning a role to a user.

The basic idea regarding risk assessment for role assignment is as follows:

- For each user u, we assign a $level\ of\ confidence$, denoted by $CNF(u)$.
- For each role R, we calculate the $minimum\ \ level\ \ of\ \ confidence$ required for the role assignment, denoted by $MLC(R)$.
- Then the $risk\ value\ 0 \leq rv(u,R) \leq 1$ for role assignment, can be calculated by the following formula:

$$rv(u,R) = \begin{cases} 0, \ if\ \ CNF(u) \geq MLC(R) \\ 1 - \frac{CNF(u)}{MLC(R)}, \ \ otherwise \end{cases}$$

Note that, in the risk computation formula above, we have assumed that the levels of confidence for users and the minimum levels of confidence required for roles should range over the same domain, such as the domain [0..3] (assuming, for example, that the domain used for expressing security levels in a system is, unclassified = 0, restricted = 1, secret = 2, top secret = 3). Therefore, we always have $0 \leq CNF(u) \leq 3$ and $0 \leq MLC(R) \leq 3$. Generally, there is domain $D = \{0, \ldots, k\}$, such that, for any $u \in \mathcal{U}$ and any $R \in \mathcal{R}$, we have $0 \leq CNF(u) \leq k$ and $0 \leq MLC(R) \leq k$.

Consider an example. Suppose that, in a file management system, reading files is considered to have security level 2 and writing files is considered to have security level 3. Administrators should have both permissions "$read\ \ files$" and "$write\ \ files$". Then the system may assign $MLC(admin) = 3$. Thus, if $CNF(lisa) = 2$, then $rv(lisa, admin)$ is 0.33; if $CNF(lisa) = 3$, then $rv(lisa, admin)$ is 0, which means there is no risk in assigning Lisa the role of administrator.

Thus, based on the above formula, in order to obtain the risk value $rv(u,R)$, the key is to calculate $MLC(R)$ for each role R. That is, we have to define a method for calculating the MLC of each role. We will use the poset-based modeling technique discussed below.

In many access control models, such as those in [7], [8], [9], a common method adopted is: subjects and objects are assigned certain security levels or given appropriate classifications; any subject can access an object only if the subject is at the same security level as the object or higher. Security levels or classification systems can be represented as partial ordering relations (posets).

As mentioned, in our method, permission is defined as a pair consisting of an action and an object, and a role is a set of such permissions. We consider both the set of actions and the set of objects as partially ordered. This ordering captures the criticality of the actions and the importance of objects. Further we compute a partial ordering on roles, i.e. a poset on set of pairs (action, object). Then, from the latter, we can obtain the minimum level of confidence (MLC) required for the role.

Let $(\mathcal{A} = \{a_i \mid i = 0, 1 \ldots, n\}, \sqsubseteq_a)$ and $(\mathcal{O} = \{o_i \mid i = 0, 1 \ldots, m\}, \sqsubseteq_o)$ be partially ordered sets of actions and objects, respectively. The relation $a' \sqsubseteq_a a$ means action a' is *less critical* than action a. Similarly, $o' \sqsubseteq_o o$ means object o' is *less important* than object o.

For example, let $a_1 = modify$, $a_2 = write$, $a_3 = move$, and $a_4 = read$. We can consider a set of actions to be ordered according to the perceived criticality of the actions. For example the action "modify" is considered to be more critical than actions "write" and "move", and the action "read" to be less critical than actions "write" as well as the action "move". However, "write" and "move" have no such relation. These facts are described by the following relations $a_2 \sqsubseteq_a a_1$, $a_3 \sqsubseteq_a a_1$, $a_4 \sqsubseteq_a a_2$, and $a_4 \sqsubseteq_a a_3$, but a_2 and a_3 are not comparable. Figure 1 presents partial orderings for a system containing the set of actions $\{a_1, a_2, a_3, a_4\}$ and the set of objects $\{o_1, o_2, o_3, o_4\}$.

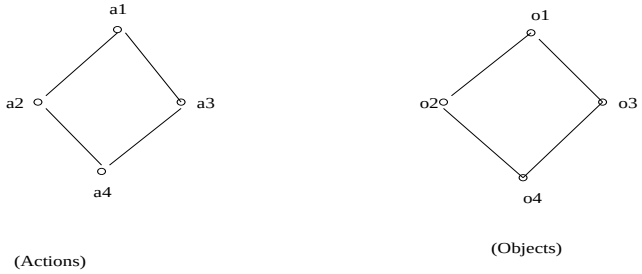

Figure 1. Actions and objects

$\mathcal{A} \times \mathcal{O}$ is the set of all permissions, we call it the *global permission set*. Based on the posets $(\mathcal{A}, \sqsubseteq_a)$ and $(\mathcal{O}, \sqsubseteq_o)$, we deduce an ordering relation $\sqsubseteq_{ao}$ as follows:

$$(a', o') \sqsubseteq_{ao} (a, o) \leftrightarrow a' \sqsubseteq_a a \wedge o' \sqsubseteq_o o$$

where $(a', o') \sqsubseteq_{ao} (a, o)$ means permission (a', o') is less critical than permission (a, o).

According to Definition 1, a role R is a subset of $\mathcal{A}\times \mathcal{O}$, i.e., a subset of the global permission set. Therefore, for any role R, $(R, \sqsubseteq_{ao})$ is also a poset. In Figure 2, we show the global permission set and two roles R1 and R2 of the system of Figure 1.

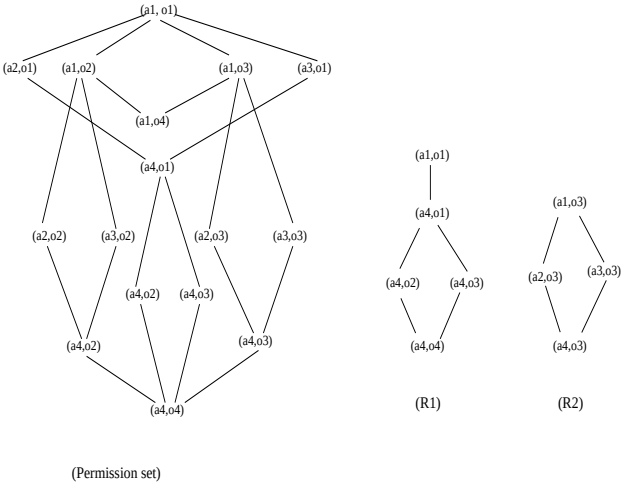

Figure 2. The global set and examples of roles

In order to calculate the MLC of a role, we introduce first the following definitions:

- A *chain* in a role R is a subset C of R having a total ordering, i.e. for all $(a, o), (a', o') \in C$, $(a, o) \sqsubseteq_{ao} (a', o')$ or $(a', o') \sqsubseteq_{ao} (a, o)$. So, all nodes in a chain C are comparable. Thus, a chain in a role can be represented by a path with directed edges between adjacent nodes.
- The length of a chain is the number of directed edges connecting two nodes in the chain. Let $l(C)$ be the length of chain C, and $n(C)$ be the number of nodes of chain C, then $l(C) = n(C) - 1$.

For example, in Figure 2, ($R1$), (a4, o4) is a shortest chain that contains only one node and its length is 0; (a4,o4)–(a4,o2) is a chain that contains two nodes, its length is 1; and (a4,o4)–(a4,o2)–(a4,o1) is a chain that contains three nodes, and its length is 2.

Let $\mathcal{C}_R$ be the set of all chains in a role $(R, \sqsubseteq_{ao})$. We define $MLC(R)$ to be the length of the longest chain in R, i.e.:

$$MLC(R) = max\{l(C)|C \in C_R\}$$

For instance, for roles described in Figure 2, we have $MLC(R1) = 3$ and $MLC(R2) = 2$.

This formula is based on the assumption that all directed edges connecting two adjacent nodes have an equal weight of 1 however, in some applications it may be necessary to assign different weights to different edges based on importance considerations.

Delegation risk is also a main issue in risk analysis. Consider the following scenario: in a company, Bob, a group head, is required to attend a meeting today, which is classified to be security level 3. Bob has no time to attend, so he asks Lisa to attend the meeting for him. However, Lisa holds only security level 2. Then this delegation may be risky.

Any role based access control system may involve such delegations, and should have appropriate formulas for calculating the risk of delegation. Different applications may have different formulas for this purpose. Here, we present a generic formula as follows: let $del\_rv(u_1, u_2)$ be the risk value of user u_1 delegating to user u_2 one of his permissions. The risk of a delegation depends on the level of confidence of the delegator and the delegatee. More formally, we may have:

$$del\_rv(u_1, u_2) = \begin{cases} 0, \ if \ CNF(u_1) \geq CNF(u_2) \\ 1 - \frac{CNF(u_1)}{CNF(u_2)}, \ otherwise \end{cases}$$

As shown in next section, we can see that the risk analysis functions defined above can play an important role in the decision making process for RBAC systems.

IV. Reasoning about RBAC Systems

Logical inference rules will be used to build a theory for reasoning and decision making in RBAC systems. We write $\varepsilon \vdash c$, where ε is an environment, and c is a logical formula containing variables and constants in ε, to mean that context c is satisfied within ε.

The environment ε contains system's access control rules and other attributes for policy evaluation such as time, location, etc. A context is a logical formula defined by the following grammar:

$$c ::= p \mid \neg c \mid c \wedge c$$

where p is an atomic proposition.

A. Reasoning without risk assessment

We first define the following predicates:

- $permit(r, a, o, c)$: Role r has the permission to conduct action a on object o within context c.
- $holds(u, r)$: User u holds role r.

- $delegate(u_1, u_2, a, o, c)$: User u_1 delegates user u_2 to conduct action a on object o within context c.
- $user\_permit(u, a, o)$: User u is permitted to conduct action a on object o.

Now we present two rules that define an RBAC Policy Decision Point (PDP). The first rule captures direct permission and the second captures delegation. We will use the following two new relations: a relation between actions and a relation between objects, both notated $\leqslant$. We write $a \leqslant a'$ to mean that a is a subaction of a', e.g. $modify \leqslant write$. We write $o \leqslant o'$ to mean that object o is included in object o'.

(1) Permission rule: According to the definition of the RBAC model, $PA \subseteq \mathcal{R} \times \mathcal{P}$, and $\mathcal{P} \subseteq \mathcal{A} \times \mathcal{O}$. Therefore, the elements of PA are of the form of $(r, (a, o))$ or simply, (r, a, o), where $r \in \mathcal{R}$, $a \in \mathcal{A}$, and $o \in \mathcal{O}$. Thus, (r, a, o) is a permission assigned to role r. Further, we extend the permission element with c, a context. The permission rule is then formalized as follows:

$$\frac{permit(r, a', o', c) \in P(\varepsilon),\ \varepsilon \vdash c,\ holds(u, r),\ a \leqslant a',\ o \leqslant o'}{user\_permit(u, a, o)}$$

where ε is an environment, and $P(\varepsilon)$ is the set of permissions in the environment ε. The permission rule states that if permission $permit(r, a', o', c)$ is in $P(\varepsilon)$, c is satisfied in ε, user u holds role r then for any subaction a of a' ($a \leqslant a'$) and subobject o of o' ($o \leqslant o'$), user u is permitted to conduct action a on object o.

Example 1.
If we have the following assumptions:
(1) $permit(trainee, modify, records, guidance) \in P(\varepsilon)$,
(2) $\varepsilon \vdash guidance$,
(3) $holds(alice, trainee)$,
(4) $write \leqslant modify$,
(5) $notes \leqslant records$.

Here $permit(trainee, modify, records, guidance)$ means that trainees have the permission to modify records under guidance, and $\varepsilon \vdash guidance$ means that in the environment ε, $guidance$ is satisfied (i.e., guidance is available). The meaning of the other assumptions is obvious. Then, based on the permission rule, we can obtain the following conclusion:
(6) $user\_permit(alice, write, notes)$.
That is, Alice is permitted to write notes.

□

(2) Delegation rule: Delegation is written in the form $delegate(u_1, u_2, a, o, c)$, The delegation rule is formalised as follows:

$$\frac{delegate(u_1, u_2, a', o', c) \in D(\varepsilon),\ \varepsilon \vdash c,\ user\_permit(u_1, a', o'),\ a \leqslant a',\ o \leqslant o'}{user\_permit(u_2, a, o)}$$

where $D(\varepsilon)$ is the set of delegations in the environment ε and $u_1, u_2 \in \mathcal{U}$, u_1 is the delegator, u_2 is the delegatee, and c is the context. The delegation rule states that:

- If a delegation $delegate(u_1, u_2, a', o', c)$ is in D(ε), and in the environment ε, c is satisifed, user u_1 has the permission (a', o') then for any sub-action a of a' and sub-object o of o', user u_1 can delegate to user u_2 to conduct action a on object o.

The permission rule and the delegation rule define a Policy Decision Point (PDP) for RBAC systems. Then with these rules, we can reason about permissions assignments in RBAC systems.

B. Reasoning in RBAC^R model with risk assessment

Risk is not explicitly included in the usual RBAC model, so in the permission and delegation rules presented above we did not consider the effects of risk. In the RBAC^R model, we add a risk parameter in the rules. In the sequel, we use the predicate $permit\_with\_risk(u, a, o, rv)$ to express the fact that user u is permitted to conduct action a on object o with a risk value rv. Furthermore, we use $risk\_threshold(\varepsilon, a, o)$ to specify a threshold value for the risk. This value is used to determine if the risk is acceptable or not for the PDP. Note that, we cannot give a general definition for this function. Its definition is rather related to specific access control applications (banking, hospital, etc.). For instance, in a banking application, the threshold may be higher if economic indicators are good (described in the ε parameter) and the requested loan (parameters a and o) is not high.

(3) Permission rule with risk assessment: The permission rule with risk assessment is formalised as follows:

$$\frac{permit(r, a', o', c) \in P(\varepsilon), \varepsilon \vdash c, holds(u, r), \\ rv(u, r) \leq risk\_threshold(\varepsilon, a, o), a \leqslant a', o \leqslant o'}{permit\_with\_risk(u, a, o, rv(u, r))}$$

This rule is the same as the standard permission rule except that a user u is permitted to conduct action a on object o with risk value $rv(u, r)$ only if this value is below a threshold given by the function $risk\_threshold(\varepsilon, a, o)$.

(4) Delegation rule with risk assessment: The delegation rule with risk assessment is formalised as follows:

$$\frac{delegate(u_1, u_2, a', o', c) \in D(\varepsilon),\ \varepsilon \vdash c, \\ permit\_with\_risk(u_1, a', o', t), \\ t + del\_risk(u_1, u_2) \leq risk\_threshold(\varepsilon, a, o), \\ a \leqslant a',\ o \leqslant o'}{permit\_with\_risk(u_2, a, o, t + del\_risk(u_1, u_2))}$$

This rule reveals an accumulation of risk due to the delegation. Indeed, suppose that a user u_1 can perform an action a on an object o with a risk value t and that user u_1 can delegate this permission to another user u_2, then the risk value for user u_2 to perform action a on object o may be greater than t, that is, $(t + del\_risk(u_1, u_2)) > t$, depending on the confidence values of u_1 and u_2.

Now, with these rules, we can reason about permissions assignment with risk assessment.

Example 2.
We assume that:
(1) $MLC(trainee) = 2, CNF(alice) = 1.9$.
(2) $risk\_threshold(\varepsilon, write, notes) = 0.1$.
From assumptions 1, we have:
(3) $rv(alice, trainee)$ = 0.05.
From assumption 2, and 3, we have
(4) $rv(alice, trainee) \leq risk\_threshold(\varepsilon, write, notes)$
By 4 and the permission rule with risk assessment, we get:
(5) $permit\_with\_risk(alice, write, notes, 0.05)$.
The last formula means that Alice is permitted to write notes with risk value 0.05.

□

V. Implementation Issues

A prototype of the system proposed above was implemented in Prolog. This implementation contains two major modules: one is for calculating risks based on risk functions, the other is for decision making.

In this implementation, the main issues we consider are:

- How to express the relations, such as $a \sqsubseteq_a a'$ and $o \sqsubseteq_o o$, and permissions as facts in Prolog,
- How to translate inference rules to Prolog rules, and
- How to translate risk functions to Prolog rules.

For simplifying the Prolog implementation, we define a unified predicate, `lessEq(X,Y)`, to represent the relations $X \sqsubseteq_a Y$ and $X \sqsubseteq_o Y$. Note that, when using the predicate, X and Y must be the same type of variables or constants, i.e., in any instance of the predicate, both X and Y are actions or both are objects. Thus, in the Prolog program, we may have the following facts:

```
lessEq(read,modify).
lessEq(modify,modify).
lessEq(notes,records).
lessEq(records,records).
```

In order to translate risk function to Prolog rules, we may need to define some predicates that can be specifically used for this purpose. For example, with this risk function

$$rv(u, R) = \begin{cases} 0, \ if\ CNF(u) \geq MLC(R) \\ 1 - \frac{CNF(u)}{MLC(R)}, \ otherwise \end{cases}$$

we first introduce predicates, `rv(U,R,RV)`, `cnf(U,X)`, and `mlc(R,Y)`. These predicates correspond to $rv(u, R)$, $CNF(u)$, and $MLC(R)$, respectively, and their meanings are as follows: `RV` $= rv(U, R)$, `X` $= CNF(U)$, and `Y` $= MLC(R)$.

Then, the function $rv(u, R)$ can be translated into two Prolog rules:

```
rv(U,R,RV):- is_user(U),
             is_role(R),
             cnf(U,X), mlc(R,Y),
             X >= Y, RV=0.
rv(U,R,RV):- is_user(U),
             is_role(R),
             cnf(U,X), mlc(R,Y),
             Y > X, Z is X/Y,
             RV = 1 - Z.
```

When $CNF(U) \geq MLC(R)$, the first rule is applied, otherwise, the second one is applied. Further, with the risk function, we also need a rule with head "mlc(R,Y)", which is applied for calculating MLC(R).

The decision rules in the Prolog program are directly translated from the rules we have proposed in Section 4. However, to obtain the appropriate Prolog rules, we need to fix ε, the environment, for a practical application by defining a state where each context has a certain truth value. For example, if in the environment ε we have $permit(r, a, o, c) \in P(\varepsilon)$, and $\varepsilon \vdash c$, then at ε we definitely have that both $permit(r, a, o, c)$ and $exists(c)$ are true. Assume that r, a, o, c represents "admin", "modify", "records" and "guidance" respectively, and $exists(c)$ may represent "guidance is available", then in this environment, we have both $permit(ass\_admin, modify, records, guidance)$ and $guidance\ is\ available$ are true. Thus, in the environment ε, the permission rule with risk can be expressed as follows:

$$permit(r, a', o', c) \wedge exists(c) \wedge holds(u, r) \wedge (a \sqsubseteq_a a') \wedge (o \sqsubseteq_o o') \wedge rv(u, r) \leq risk\_threshold(\varepsilon, a, o) \rightarrow user\_permit(u, a, o).$$

Translating the rule directly, we get the following Prolog decision rule:

```
permitInRisk(U,A2,O2,RV) :-
            p(R,A1,O1,c), exists(c),
            holds(U,R), lessEq(A2,A1),
            lessEq(O2,O1), rv(U,R,RV),
            thred(A2,O2,Threshold),
            RV <= Threshold.
```

VI. Conclusion

Risk analysis is an important issue for access control systems. In this paper, we have proposed a role based access control model with risk, called RBAC^R. Within this model, we have proposed a method for risk analysis, which enables systems to evaluate the risks associated with access requests. This model shows how assessing risk for decision making could be done in principle in such systems.

Future work will include refining the partial orders to include more sophisticated risk measures. Dynamic aspects of security requirements associated with the RBAC^R model will also be taken into consideration. Another interesting aspect is the investigation of management issues of RBAC^R policies.

Acknowledgement

This research has been funded in part by grants from PROMPT Québec and from CA Labs.

References

[1] M. A. Al-Kahtani and R. S. Sandhu. A model for attribute-based user-role assignment. In *ACSAC*, pages 353–364, 2002.

[2] B. Aziz, S. N. Foley, J. Herbert, and G. Swart. Reconfiguring role based access control policies using risk semantics. *J. High Speed Networks*, 15(3):261–273, 2006.

[3] D. E. Bell and L. J. LaPadula. Secure computer system: Unified exposition and multics interpretation. In *Technical Report MTR-2997*, The MITRE Corporation, Bedford, Massachusetts, 1976.

[4] N. Dimmock, A. Belokosztolszki, D. M. Eyers, J. Bacon, and K. Moody. Using trust and risk in role-based access control policies. In *SACMAT*, pages 156–162, 2004.

[5] S. Kondo, M. Iwaihara, M. Yoshikawa, and M. Torato. Extending RBAC for large enterprises and its quantitative risk evaluation. In *II3E*, pages 99–112, 2008.

[6] N. Nissanke and E. J. Khayat. Risk based security analysis of permissions in rbac. In *WOSIS*, pages 332–341, 2004.

[7] S. A. Obiedkov, D. G. Kourie, and J. H. P. Eloff. On lattices in access control models. In *ICCS*, pages 374–387, 2006.

[8] R. S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, 1993.

[9] R. S. Sandhu. Role hierarchies and constraints for lattice-based access controls. In *ESORICS*, pages 65–79, 1996.

[10] R. S. Sandhu and V. Bhamidipati. An oracle implementation of the PRA97 model for permission-role assignment. In *ACM Workshop on Role-Based Access Control*, pages 13–21, 1998.

[11] H. Takabi, M. Amini, and R. Jalili. Trust-based user-role assignment in role-based access control. In *AICCSA*, pages 807–814, 2007.