A Formal Model For Role-Based Access Control with Constraints

Luigi Giuri, Pietro Iglio
Fondazione Ugo Bordoni
via B. Castiglione, 59 - 00142 - Roma, Italy
{giuri, iglio}@fub.it

Abstract

The aim of this work is to give a formal foundation to the notion of role-based access control through the introduction of a new model and the formal specification of its semantics.

The proposed model takes into account all the main topics currently under discussion in this area, including constraints and separation of duties. Moreover, it is suitable both for conceptual design purpose and direct implementation within real systems.

1. Introduction

Role-based access control is probably the most interesting and promising technique recently proposed for design and implementation of modern system security policies. It is based on the common practice in organisations of assigning duties and responsibility to the employees on the basis of their role within the organisation itself.

In the last few years, this fact has assumed a great importance for the research that has been conducted in the computer security area. Much work has been done in the definition and implementation of models for role-based access control. In this way the computer system security policy resembles much more the corporate security policy and all the other higher-level security policies by which it depends on.

Reducing the divergence between these policies causes an increase in security comprehensibility and manageability for the entire organisation, that is an improvement of the global degree of security. Moreover, enforcing a security policy through role-based access controls permits to achieve particular security objectives, such as least privilege and separation of duty, that are considered as very interesting for commercial and civilian government organisations [CLA87, FER95].

Thus, the objective of the past and present research is to define practical models for role-based access control that allows user behaviour within a computer system to be related to his or her duties and responsibility within the organisation.

For example, such models have been developed for relational databases [BAL90], object-oriented databases [HU93, HU94, NYA93, RAB91, TIN92], and secure on-line transaction processing systems [SMI94].

Recently, [SAN95] has proposed a framework that provides a systematic approach to understand role-based access control, and to categorise its implementation in different systems. Within this framework, two fundamental aspects of role-based access control models are highlighted, namely role hierarchies and constraints.

In the model proposed by [GIU95a], the focus is essentially on separation of duties, providing a way to express implicit mutual exclusion constraints for roles within the role hierarchy. However, such model does not enable to specify different kinds of application-specific constraints.

This work describes a new model for role-based access control that addresses such problem, providing capabilities to include generic constraints in the design of role-based security system policies.

Section 2 introduces the concepts that will be used in the rest of the paper, using the model proposed by [BAL90] and the model proposed by [GIU95a] as starting points. Section 3 extends the latter, in order to permit the specification of constraints on roles. Section 4 provides a formal semantics for the fundamental concept of selection, introduced in the previous sections. Section 5 discusses some useful properties that can be used in real implementations in order to simplify the selection process. Finally, we present our conclusions in section 7.

1063-6900/96 $5.00 © 1996 IEEE

2. Basic Models

This section provides the basic concepts for role-based access control using the models presented in [BAL90] and [GIU95a] as groundwork. Before discussing the models proposed in such works, we introduce some definitions that will be used in the rest of this paper.

A *privilege* is the right to perform some operation. Privileges are often partitioned in object privileges and system privileges. An *object privilege* is the right to perform a specific operation on a particular object. For example, object privileges on a database table include the ability to SELECT, INSERT, UPDATE, and DELETE data from that table. A *system privilege* is the right to perform a specific operation in the system. For example, system privileges on a database include the ability to create database user names (CREATE USER), tables (CREATE TABLE), and so on.

We do not define a particular syntax for privileges, as this is strictly related to the specific system considered. The syntax defined for the remaining concepts, introduced in the rest of this document, is fairly general and does not make assumptions about the underlying system.

A *protection domain* is the set of privileges which identifies the set of all operations that could be performed by a subject at a given time:

$\{priv_1, ..., priv_n\}$

The protection domain that controls the user behaviour during a particular session is the *active protection domain*.

2.1 NPD Model

According to [BAL90], a role is defined by the concept of *named protection domain* (NPD). In this model, a role is an explicit (i.e. named) representation of a collection of privileges which are defined and used by system administrators and users. A role can be defined by including privileges or other roles. The role definition is represented by the following syntax:

$r = \mathbf{role}(priv_1, ..., priv_n, r_1, ..., r_m)$

In the above expression, r represents the name assigned to the new role, $priv_1$, ..., $priv_n$ are the *direct privileges* of *role* (i.e., privileges directly assigned to r), and r_1, ..., r_m are the names of the *direct subroles* of r. Direct privileges and/or direct subroles can eventually be absent from a role definition. The *subroles* of r are constituted by r and the subroles of its direct subroles.

With respect to the above definitions, the protection domain corresponding to a role r is composed by its direct privileges and the privileges of its subroles.

Loops are not allowed in the definition of roles. This means that the (finite) set of roles within a system has a hierarchical structure, where the bottom part consist of the roles having no subroles.

A role can be *activated*, triggering the activation of the corresponding protection domain. Generally, roles can be assigned to users that can activate such roles and their subroles. For example, the Oracle DBMS [ORA92] implements the NPD model according to this policy. However, different policies for the assignment and the activation of roles can be adopted. For example, a role could be assigned to a transaction in order to restrict the user access to some information (i.e., the activation of a specific protection domain) only when he or she is performing a particular operation.

A graphical representation of role definitions may be obtained using a graph (which is called *privilege graph*) where the nodes represent users, roles, and privileges, and the edges represent the assignment of privileges and roles to other roles and users.

An example of privilege graph is shown in figure 1. This example represents a corporation where there are basic clerks that perform simple routine operations, and clerks that perform all the operations of basic clerks and, eventually, handle payable or receivable accounts. Moreover, clerks cannot handle both payable or receivable accounts at the same time. The same situation will be considered in other examples that will be shown in the rest of this paper.

2.2 NSPD Model

An alternative definition for role has been proposed in [GIU95a]. The main advantage of this model, if compared to the NPD model, is that it permits to represent situations related to the enforcement of separation of duty requirements in a more straightforward and natural way.

In the proposed model, a *role* is defined as a *named set of protection domains* (NSPD). A set of protection domains specifies a collection of possible sets of privileges that control the user behaviour within the system:

$\{\{priv_{11},..., priv_{1i}\},..., \{priv_{n1},..., priv_{nj}\}\} = \{Pd_1,..., Pd_n\}$

At a given time, only one of these protection domains can be the active protection domain. While in the NPD model a role activation causes the activation of the corresponding protection domain, in the case of the

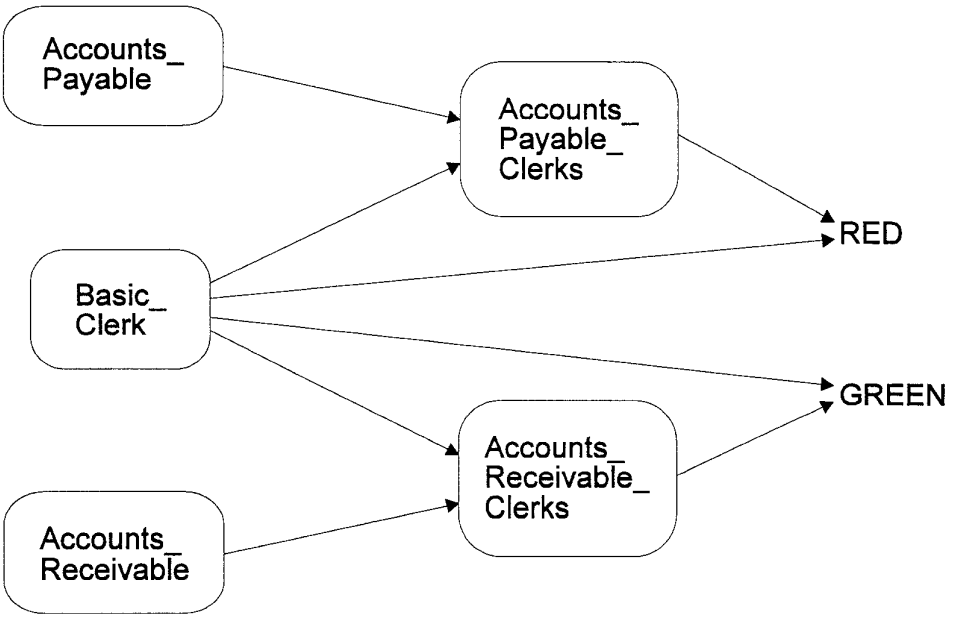

Figure 1. Example of NPDs

NSPD model it is necessary to select one of the protection domains associated to a role. How this selection is performed will be explained later in this section.

Starting from this definition, the NSPD model defines a language for the specification of roles. More precisely, the language defines two structures for the specification of sets of protection domains:

- the *and-role*, that is a set of privileges and roles simultaneously used:

  r = [**optional**] **and**($priv_1$, ..., $priv_n$, r_1, ..., r_m)

- the *or-role*, that is a set of mutually exclusive roles (only one role among r_1, ..., r_m can be used):

  r = [**optional**] **or**(r_1, ..., r_m)

If the **optional** identifier is specified, it declares an *optional role*, that is a role containing an empty protection domain. This means that users have the choice whether or not to activate that role.

As in the NPD model case, it is possible to define a graphical representation for the NSPD model through a privilege graph. An example is shown in figure 2. This example represents the same situation modelled in figure 1.

It is required that an or-role contains only other roles in order to allow the selection of the active protection domain through the name of the corresponding role. The remaining part of this section describe the syntax defined for this purpose.

A *selection* is specified with the following syntax:

$\underline{r}$ **is** r_k

where $\underline{r}$ is an or-role and r_k is one of the direct subroles that appear in the specification of r.

In the case of multiple nested or-roles, a selection must be specified for all the or-roles involved by successive selections. If there is some role that has not a selection, then the specified selections identify a set of protection domains. The interesting selections are those that identify a single protection domain, that is the enabled protection domain for a particular user session.

Moreover, in the case of use of an optional role (either or-role or and-role) a declaration of use must be stated through the following *use clause*:

use $\underline{r}$

Finally, a *multiple selection* for a role r consists of a set of selections and use clauses. The syntax for a multiple selection is:

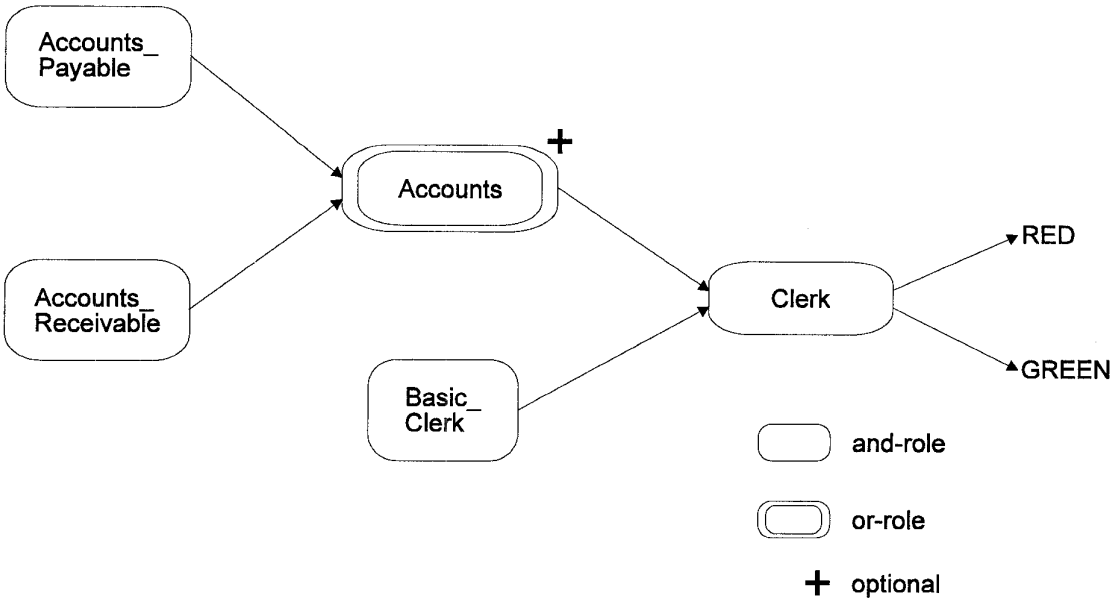

Figure 2. Example of roles as NSPDs

select($\underline{r}_1$ **is** sr_1, ..., $\underline{r}_p$ **is** sr_p, **use** $\underline{r}_1$, ..., **use** $\underline{r}_q$)

where each role in a single selection or use clause is either r or a subrole of r.

In the privilege graph, a multiple selection for a role identifies a subgraph of the graph corresponding to that role.

3. Adding Constraints

One important aspect of modern security policies, that is not considered in the NSPD model, is the capability of specifying constraints. Constraints are used to make some protection domains activatable or not, depending on external conditions. For example, a system security policy might allow the activation of a particular role only from Monday to Friday and/or only from terminals located in a trusted environment. Generally, constraints provide more flexibility to the security system as they allow the control of user behaviour according to some conditions that can be verified only at run-time.

Constraints can be of different kinds: temporal constraints (e.g., a given role can be activated only during working days), system-dependent constraints (e.g., the terminal from which a user can log in), data-dependent constraints (e.g., the result of a database query), and so on.

In order to provide the capability of defining constraints on roles, we define a new model that is an extension of the NSPD model. It is important to highlight that, in the model introduced, constraints are not necessary to achieve separation of duties, which is already realised by constructs of the NSPD model.

In this section we introduce a new semantic interpretation for the concept of role and we provide an extension to the previously introduced syntactic constructs so that constraints can be easily specified. Then, we provide formal semantics and properties related to the new concepts.

A *constrained protection domain* is a protection domain with an associated constraint c:

$<c, \{priv_1, ..., priv_n\}>$

where c could be considered as a formula of the First Order Logic. However, implementations could provide a restricted set of formulas for efficiency reasons. For example, an implementation within a SQL database management system could only allow constraints that are accepted in the `WHERE` clause of the `SELECT` statement.

A constrained protection domain is *activatable* only if the corresponding constraint is satisfied. Note that a hypothetical system implementing constrained protection domains could adopt different policies

regarding the time at which the constraint is evaluated. Such issue will be discussed in section 6.

In the new model, a role is defined as a *named set of constrained protection domains* (NSCPD). A set of constrained protection domains specifies a collection of possible sets of privileges that control the user behaviour within the system:

$\{<c_1, \{priv_{11},..., priv_{1i}\}>,...,<c_n, \{priv_{n1},..., priv_{nj}\}>\} = \{<c_1, Pd_1>,..., <c_n, Pd_n>\}$

Similarly to the NSPD model, only one of the constrained protection domains can be active at a given time. Note that this does not mean that only one of $c_1, ..., c_n$ is true. However, the active constrained protection domain must belong to the set of constrained protection domains whose corresponding constraint is true. If there are no satisfied constraints, then the role cannot be activated.

In order to specify roles based on the NSCPD model, we must extend the previously defined specification language to include constraints. The syntactic constructs are extended so that a role definition can eventually include constraint specifications for the role itself (*role condition*) and for its direct subroles (*subrole condition*). As far as the graphical representation is concerned, this means that it is possible to associate constraints to nodes, (role conditions) and edges (subrole conditions).

In the NSCPD model, the *and-role* is a set of privileges and roles that can be simultaneously used, provided that the corresponding constraints are satisfied:

r = [**optional**] **and**($priv_1$, ..., $priv_n$, r_1 [**when** c_1], ..., r_m [**when** c_m]) [**when** c]

In this definition, c is a constraint that must be satisfied in order to activate r. If c is satisfied, then the constraints c_1, ..., c_m must be satisfied in order to activate the corresponding direct subroles.

An *or-role* is a set of mutually exclusive roles with an associated constraint:

r = [**optional**] **or**(r_1 [**when** c_1],..., r_m [**when** c_m]) [**when** c]

Similarly to the and-role case, c is a constraint that must be satisfied in order to activate r. If c is satisfied, then a unique direct subrole r_i, such that c_i is satisfied, must be activated. If there is no satisfied c_i, then no direct subrole is activated.

The semantic of the **optional** identifier is the same as in the NSPD model.

Figure 3 shows an example of privilege graph for the NSCPD model. This example represents the same situation modelled in figures 1 and 2, adding the following restrictions: clerks that must process payable accounts can perform this task only from a specific terminal tty1; clerks that must process payable accounts can perform this task only from terminal tty5; every clerk can operate only from Monday to Friday.

3.1 Formal Semantics

In this section we provide a formal semantics for the NSCPD model. More precisely, we define an interpretation for the syntactic constructs that have been previously defined. The interpretation is provided by a function *SCPD* that associates the appropriate meaning, corresponding to a set of constrained protection domains, to a syntactic form.

The *SCPD* function is differently defined depending on the role type. If r is a non optional and-role, *SCPD* is defined as:

$$SCPD(r) = \{<c, \{priv_1, ..., priv_n\}>\} \otimes ((c \wedge c_1) \Lambda SCPD(r_1)) \otimes ... \otimes ((c \wedge c_m) \Lambda SCPD(r_m))$$

where the symbol $\otimes$ represents the *SCPD-product* operator, defined as follows:

$$\{<c_{11},Pd_{11}>,...,<c_{1n},Pd_{1n}>\} \otimes \{<c_{21},Pd_{21}>,...,<c_{2m},Pd_{2m}>\}$$
$$= \{<c_{1i} \wedge c_{2j}, Pd_{1i} \cup Pd_{2j}> \mid 1 \le i \le n, 1 \le j \le m\}$$
$$\cup \{<c_{1i} \wedge (\neg c_{21} \wedge ... \wedge \neg c_{2m}), Pd_{1i}> \mid 1 \le i \le n\}$$
$$\cup \{<(\neg c_{11} \wedge ... \wedge \neg c_{1n}) \wedge c_{2j}, Pd_{2j}> \mid 1 \le j \le m\}$$

and the symbol Λ represents the *SCPD-and* operator, defined as follows:

$$c \Lambda \{<c_1, Pd_1>, ..., <c_n, Pd_n>\} = \{<c \wedge c_1, Pd_1>, ..., <c \wedge c_n, Pd_n>\}$$

Informally, the SCPD-product operator generates the set of all the possible choices for two set of constrained protection domains that are simultaneously used, while the SCPD-and operator simply add a constraint to all the elements contained in a set of constrained protection domains. Since the SCPD-product operator is associative (the proof is omitted), the *SCPD* function is well-defined.

If r is a non optional or-role, *SCPD* is defined as:

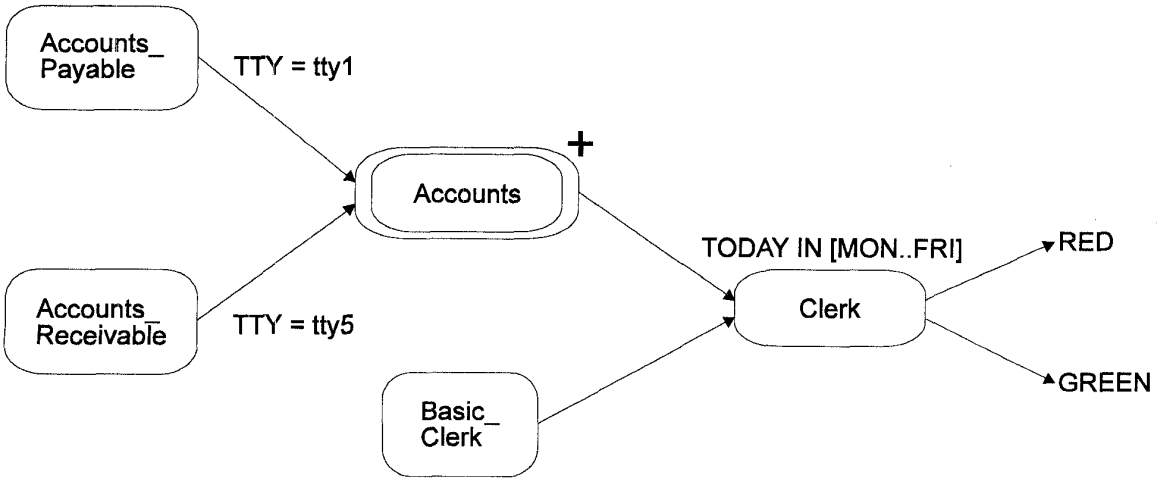

Figure 3. Example of roles as NSCPDs

$SCPD(r) =$
$((c \wedge c_1) \Lambda\ SCPD(r_1)) \cup ... \cup ((c \wedge c_m) \Lambda\ SCPD(r_m))$

The optional roles are treated adding <T, { }> to the set of constrained protection domains that results considering the role as non optional, where T denotes the logical value "true".

If r is an optional and-role, *SCPD* is defined as follows:

$$SCPD(r) = \{<T, \{\}>\} \cup (\{<c, \{priv_1, ..., priv_n\}>\} \otimes ((c \wedge c_1) \Lambda\ SCPD(r_1)) \otimes ... \otimes ((c \wedge c_m) \Lambda\ SCPD(r_m)))$$

If r is an optional or-role, *SCPD* is defined as follows:

$$SCPD(r) = \{<T, \{\}>\} \cup ((c \wedge c_1) \Lambda\ SCPD(r_1)) \cup ... \cup ((c \wedge c_m) \Lambda\ SCPD(r_m))$$

3.2 Definitions

In this section we introduce some definitions that are useful to highlight some properties of roles in the NSCPD model.

In the subsequent definitions we consider a generic or-role r of the following form:

r = [**optional**] **or**(r_1 [**when** c_1],..., r_m [**when** c_m])

[**when** c]

An or-role r is *complete* if the following expression is always true:

$c_1 \vee ... \vee c_m$

The above definition means that, at a given time, there is at least one activatable direct subrole for r.

An or-role r is *self-specified* if the following expression is always true:

$\forall\ i, j \in [1, m]: i \neq j \rightarrow (c_i \rightarrow \neg c_j)$

The above definition means that, at a given time, there is at most one direct subrole of r that can be activated.

An or-role r is *auto-specified* if it is complete and self-specified. This property means that, at a given time, there is one and only one activatable direct subrole for r, i.e. the underlying system can automatically decide which subrole must be activated for r.

The above definitions consider all the possible situations modelled by the logical language used for the formulas. However, many of these situations could be useless because some of them could be not interesting or could be meaningless in a particular environment. In such case, it is necessary to restrict the target of the analysis in order to make useful the analysis itself. Therefore, the following definitions restrict the evaluation of constraints to a subset of all the possible situations through the use of a logical formula.

An or-role r is *complete with respect to a formula* φ if the following expression is always true:

$$\varphi \rightarrow (c_1 \vee ... \vee c_m)$$

An or-role r is *self-specified with respect to a formula* φ if the following expression is always true:

$$\varphi \rightarrow (\forall\, i, j \in [1, m]: i \neq j \rightarrow (c_i \rightarrow \neg c_j))$$

An or-role r is *auto-specified with respect to a formula* φ if it is complete and self-specified with respect to φ.

In practical cases, it is more interesting to consider roles that are auto-specified with respect to a particular formula, such that it is possible to have role-scheme that are concise and simple to manage.

4. Selection

Similarly to the NSPD case, the concept of selection is essential for the NSCPD model in order to specify which roles must be considered when a given role is activated, so that an activatable constrained protection domains can be chosen for activation.

In this section we provide definitions and a formal semantics for the multiple selection construct. The syntax used for the NSPD model does not change in the NSCPD model. In subsections that follow we will consider a generic multiple selection *msel* for a role r, defined as:

select($\underline{r}_1$ **is** sr_1, ..., $\underline{r}_p$ **is** sr_p, **use** $\underline{r}_1$, ..., **use** $\underline{r}_q$)

where $\underline{r}_1$, ..., $\underline{r}_p$ are or-roles and $\underline{r}_1$, ..., $\underline{r}_q$ are optional-roles, according to the definitions given in section 2.2.

4.1 Definitions

In this section we introduce some definitions that can be used to identify and to reject meaningless multiple selections. Moreover, these definitions can be used to simplify a multiple selection for a given role by using constraints.

Given a role r, a multiple selection *msel* for r, and a subrole r' of r, r' is *reachable* from r with respect to *msel* if one of the following properties apply:

- $r' = r$;
- r' is a direct subrole of a reachable and-role r'';
- r' is a direct subrole of a reachable or-role r'' and r'' **is** $r' \in$ *msel*.

Furthermore, if r' is optional then **use** r' must belong to *msel*.

Informally, the meaning of the above definition is that there is a path from r to r' in the subgraph corresponding to *msel*.

A multiple selection *msel* for a role r is *connected* if, for each role r' such that r' **is** $r'' \in$ *msel* or **use** $r' \in$ *msel*, r' is reachable from r with respect to *msel*.

A multiple selection *msel* for a role r is *correct* if it is connected and, for each or-role r' that is a subrole of r, there is at most one selection in *msel* of the form: r' **is** r''.

A multiple selection *msel* for a role r is *complete* if it is connected and, for each or-role r' that is a subrole of r, there is at least one selection in *msel* of the form: r' **is** r''.

A multiple selection for a role r is *valid* if and only if it is correct and complete, i.e. for each role r' in the subgraph starting from the role r and identified by the multiple selection, if r' is an or-role then one and only one of its subroles is selected.

In order to show the intuitive meaning of the above definitions, we provide some examples. All the following examples refer to the privilege graph shown in figure 4.

If we consider the multiple selection:

msel = (r2 **is** r5, r5 **is** r7)

then the subrole r7 is reachable, but r6 is not. Note that *msel* is valid. Multiple selection (r2 **is** r5) is correct but not complete, as for subrole r5 there is no selection of the form: r5 **is** r''. Finally, the multiple selection (r2 **is** r4, r2 **is** r5) is an example of multiple selection that is not correct.

Note that the definitions provided in this section also apply to the NSPD model, as constraints do not appear inside their definitions.

4.2 Formal Semantics

Once we have introduced the above definitions, we can provide a semantic interpretation for a multiple selection. The interpretation is defined through the function *Sel* that associates a given role and a correct multiple selection for that role to the corresponding set of constrained protection domains.

If r is a non optional and-role, *Sel* is defined as follows:

$$\begin{aligned} Sel(r, msel) = \{<c, \{priv_1, ..., priv_n\}>\} \\ \otimes ((c \wedge c_1) \Lambda\, Sel(r_1, msel)) \\ ... \\ \otimes\ ((c \wedge c_m) \Lambda\, Sel(r_m, msel)) \end{aligned}$$

If r is a non optional or-role, *Sel* is defined as follows:

$$Sel(r, msel) =$$
$$(c \wedge c') \wedge Sel(sr_i, msel - \{r \textbf{ is } sr_i\}) \quad \text{if } \exists\, \underline{r_i} \textbf{ is } sr_i \in msel \mid r = \underline{r_i},$$

$$(c \wedge c_1) \wedge Sel(r_1, msel) \cup \ldots \cup (c \wedge c_m) \wedge Sel(r_m, msel) \quad \text{otherwise.}$$

where c' is the constraint for the subrole sr_i in the definition of r, and $c_1 \ldots c_m$ are the constraints for the corresponding subroles $r_1 \ldots r_m$ in the definition of r.

Moreover, if r is an optional role, $Sel(r, msel)$ is defined as $\{<T, \{\}>\}$ if **use** $r \notin msel$, otherwise it is defined as in the corresponding non optional case.

Informally, the interpretation of a multiple selection for a role gives the set of constrained protection domains associated to that role restricted according to the choices specified by the multiple selection. The correctness of the multiple selection ensures that the function *Sel* is well-defined and does not return useless results.

Note that if a multiple selection *msel* for a given role is valid, then the *Sel* function returns a set with a single constrained protection domain, that will be the activated one when *msel* is provided for the activation of the related role.

5. Automating Selections

In this section we put together properties of roles and multiple selections in the NSCPD in order to describe a method to permit automation of the selection process.

5.1 Definitions

This section provides some definitions that are strictly related to the ones shown in section 4.1, but that take advantage of role constraints in order to simplify multiple selections for roles, as we will explain in the next section.

Given a role r, a multiple selection *msel* for r, a logical formula φ, and a subrole r' of r with role condition c', r' is *logically reachable* from r with respect to *msel* and φ if $\varphi \rightarrow c'$ is true, and one of the following properties applies:

- $r' = r$;
- r' **when** ec' is a direct subrole of a logically reachable and-role r'' and $\varphi \rightarrow ec'$ is true;
- r' **when** ec' is a direct subrole of a logically reachable or-role r'', $\varphi \rightarrow ec'$ is true, and at least one of the following conditions applies:
  - r'' **is** $r' \in msel$;
  - r'' is auto-specified with respect to φ.

Furthermore, if r' is optional then **use** r' must belong to *msel*.

The above definition is very similar to the definition of reachable role. The difference is that the former takes advantage of the auto-specified property, i.e. it is not required that there are explicit selections for auto-specified roles.

A multiple selection *msel* for r is *logically connected* with respect to φ if, for each role r' such that r' **is** $r'' \in msel$ or **use** $r' \in msel$, r' is logically reachable from r with respect to *msel* and φ.

A multiple selection *msel* for a role r is *logically correct* with respect to φ if it is logically connected with respect to φ and, for each or-role r' that is a subrole of r, there is at most one selection in *msel* of the form: r' **is** r''.

A multiple selection *msel* for a role r is *logically complete* with respect to φ if it is logically connected with respect to φ and, for each or-role r' that is a subrole of r, r' is auto-specified or there is at least one selection in *msel* of the form: r' **is** $role_i$.

A multiple selection is for a role r is *logically valid* with respect to φ if and only if it is logically correct and logically complete with respect to φ, i.e. for each role r' in the subgraph starting from the role r and identified by the multiple selection, if r' is an or-role then exactly one of its subroles is selected (either by an explicitly selection or by an implicit one, if the or-role is auto-specified) and each constraint corresponding to nodes and edges in the subgraph is implied by φ.

In order to show the intuitive meaning of the above definitions, we provide some examples. All the following examples refer to the privilege graph shown in figure 4, where we assume that there is a formula φ such that r2 is auto-specified with respect to φ and $\varphi \rightarrow c_2$. In this situation, given an empty multiple selection:

$msel = ()$

the subrole r5 is logically reachable with respect to *msel* and φ, but is not reachable with respect to *msel*.

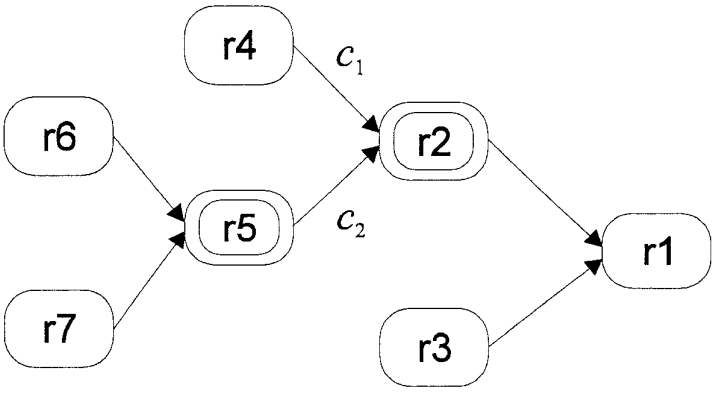

Figure 4. Example of roles with constraints

If we consider the same φ, the multiple selection (r5 **is** r7) is logically valid, since r5 is auto-specified, but not valid.

5.2 Formal Semantics

The last definition we want to introduce concerns the interpretation function *AutoSel*. Such function is similar to *Sel*, but takes into account the state of the system to take advantage of auto-specified roles. This way, *AutoSel* produces a set of constrained protection domains starting from a simpler selection. From a practical point of view, this means that part of the multiple selection needed by *Sel* is automatically determined.

The *AutoSel* function is defined for a multiple selection *msel* that is logically correct with respect to a formula φ, and a formula σ (corresponding to the state of the system) such that $\sigma \rightarrow \varphi$. If *r* is a non optional and-role, *AutoSel* is defined as follows:

$$AutoSel(r, msel, \sigma) = \{<c, \{priv_1, ..., priv_n\}>\} \otimes ((c \wedge c_1) \Lambda AutoSel(r_1, msel, \sigma)) \ldots \otimes ((c \wedge c_m) \Lambda AutoSel(r_m, msel, \sigma))$$

If *r* is a non optional or-role, *AutoSel* is defined as follows:

$$AutoSel(r, msel, \sigma) = (c \wedge c') \Lambda AutoSel(sr_i, msel - \{r \textbf{ is } sr_i\}, \sigma) \quad \text{if } \exists\, \underline{r_i} \textbf{ is } sr_i \in msel \mid r = \underline{r_i}$$

$$(c \wedge c_j) \Lambda AutoSel(r_j, msel - \{r \textbf{ is } r_j\}, \sigma) \quad \text{if } \sigma \rightarrow c_j \text{ and } r \text{ is auto-specified wrt } \varphi$$

$$(c \wedge c_1) \Lambda AutoSel(r_1, msel, \sigma) \cup \ldots \cup (c \wedge c_m) \Lambda AutoSel(r_m, msel, \sigma) \quad \text{otherwise.}$$

where *c'* is the constraint for the subrole sr_i in the definition of *r*, and $c_1 \ldots c_m$ are the constraints for the corresponding subroles $r_1 \ldots r_m$ in the definition of *r*.

We can show an example of application of the *AutoSel* function considering the situation represented in figure 3. In that example, role Accounts is auto-specified with respect to the following situation:

use Accounts ∈ *msel* → TTY = tty1 ∨ TTY = tty5

that is, a clerk enables the optional role Accounts only if he or she is working from terminal tty1 or tty5. It does not make sense for a clerk to try to enable role Accounts from another terminal, because in such case the system would not permit any accounting operation.

If a clerk logs in from terminal tty1, it is only required that he or she specifies the multiple selection (**use** Accounts) to automatically activate roles: Clerk, Accounts, Accounts_Payable, and Basic_Clerk. In a system not implementing the auto-selection, the corresponding multiple selection would have been: (**use** Accounts, Accounts **is** Accounts_Payable).

6. Implementation Issues

The NSCPD model is suitable for direct implementation within the current database management systems that already implement other existing role models.

In [GIU95a] is discussed how the implementation of the NSPD model can be realised without changing

the basic mechanisms for access control used in currently available database management systems.

Since the NSCPD model extends the NSPD model adding constraints, it would be necessary to realise an activation procedure that traverses the privilege graph to choose, using a valid multiple selection, only one alternative for each or-role involved, and verifies all the constraints that have been eventually associated to nodes and edges in the privilege graph, possibly taking advantage of the self-specified and auto-specified roles, as discussed before.

One important issue, that has not been discussed in the previous sections, is the time at which a constraint is evaluated. A simple implementation might evaluate constraints only when a role is activated for a user. If the constraint is no more satisfied during the session, the user protection domain would not change.

A more sophisticated solution might be implemented within an active database system. Trigger mechanisms might be used to dynamically change the user protection domain as soon as a specified constraint is no more satisfied.

Anyway, our model does not specify this issue, leaving it as an implementation choice.

7. Conclusions

This paper presented a new model for role-based access control that fulfils many of the main requirements for modern security policies. Furthermore, we have pointed out some properties that can be considered in real implementations.

The expressive power of the new model makes it suitable both for conceptual and logical design. For example, a possible work could be to study how the proposed model can be used together with a conceptual database model (like the Entity-Relationship data model). A translation procedure from the conceptual model to a logical one (for example, the relational database model with the NPD role model) could be defined to build a complete database design methodology, including the security aspects of the system. A similar work has already been performed for the NSPD model [GIU95b].

References

[BAL90] Baldwin R. W., "Naming and Grouping Privileges to Simplify Security Management in Large Databases", in Proceedings of 1990 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, May 1990.

[CLA87] Clark D. D., Wilson D. R., "A Comparison of Commercial and Military Security Policies", in Proceedings of 1987 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, April 1987.

[FER95] Ferraiolo D., Cugini J., Kuhn R., "Role-Based Access Control (RBAC): Features and Motivations", in Proceedings of 11th Annual Computer Security Application Conference, New Orleans, LA, December 13-15, 1995.

[GIU95a] Giuri L., "A New Model for Role-Based Access Control", in Proceedings of 11th Annual Computer Security Application Conference, New Orleans, LA, December 13-15, 1995.

[GIU95b] Giuri L., Iglio P., "Traduzione Automatica tra Modelli di Sicurezza Basati su Ruoli", FUB Technical Report 3B07745, December 1995 (in Italian).

[HU93] Hu M.-Y., Demurjian S. A., Ting T. C., "User-Role Based Security Profile for an Object-Oriented Design Model", in Database Security VI: Status and Prospects, North-Holland, 1993.

[HU94] Hu M.-Y., Demurjian S. A., Ting T. C., "User-Role Based Security in the ADAM Object-Oriented Design and Analyses Environment", in Proceedings of the IFIP WG 11.3 Eight Annual Working Conference on Database Security, August 1994.

[NYA93] Nyanchama M., Osborn S., "Role-Based Security, Object-Oriented Databases & Separation of Duties", ACM SIGMOD RECORD, Vol.22, No.4, December 1993.

[ORA92] *ORACLE7™ Server - SQL Language Reference Manual*, December 1992

[RAB91] Rabitti F., Bertino E., Kim W., Woelk D., "A Model of Authorization for Next-Generation Database Systems", ACM Transactions on Database Systems, Vol.16, No.1, March 1991.

[SAN95] Sandhu R. S., Coyne E. J., Feinstein H., Youman C.E., "A Family of Role-Based Access Control Models", in Small Business Innovation Research (SBIR) - Role-Based Access Control - Phase 1, Final Report, SETA Corporation, May 1995.

[SMI94] Smith-Thomas B., Chao-Yeuh W., "Implementing Role Based, Clark-Wilson Enforcement Rules in a B1 On-Line Transaction Processing System", in Proceedings of 17th National Computer Security Conference, October 1994.

[TIN92] Ting T. C., Demurjian S. A., Hu M.-Y., "Requirements, Capabilities, and Functionalities of User-Role Based Security for an Object-Oriented Design Model", in Database Security V: Status and Prospects, North-Holland, 1992.