Hierarchical secure systems: A preliminary description

Jason Crampton* and George Loizou

Department of Computer Science, Birkbeck College, University of London,
Malet Street, London, WC1E 7HX, England

e-mail: ccram01@dcs.bbk.ac.uk

29th November 2000

Abstract

We introduce the hierarchical secure systems model which is a generalisation of the Bell-LaPadula model for multi-level secure systems, and incorporates ideas from role-based access control and the theory of partially ordered sets. We show that the Bell-LaPadula, protection matrix and role-based access control models are particular instances of the hierarchical secure system model. We also demonstrate that this model provides a useful extension to the role-based access control model and obviates many of the more complicated aspects of the model. In particular, we propose an intuitively appealing administrative model for role-based access control which is more simple and more secure than existing models.

1 Introduction

Our most recent work is concerned with the application of antichains in a partially ordered set (see Section 3 and [4]) to access control modelling. We have successfully applied our methods to modelling conflict of interest policies [5] and to certain aspects of role-based access control [3]. A special case of one of our results forms part of the motivation for this work.

Our earlier work [6, 7, 15] is concerned with modelling the behaviour of a discretionary *reference monitor* (or *access control subsytem*) of a computer system using a deductive database. The motivation for this approach was the extensive evidence that the implementation of security (which typically uses a reference monitor based on a discretionary access control model) in a commercial computer system rarely corresponds to the requirements of the organisation running that system [15]. The purpose of this work is to reason about the correctness of the implementation (that is, the configuration of access control lists, say) of an abstract access control policy. The ultimate goal is to provide a theoretical basis for the development of software tools to be used by systems administrators to determine which access rights should be granted.

We model the state of the reference monitor as a set of (access right) triples $M \subseteq O \times S \times R$, where O is the set of objects, S is the set of subjects, and R is the set of access rights supported by the system. Our model is essentially the same as that of Harrison, Ruzzo and Ullman [10] with $(o, s, r) \in M$ if, and only if, $r \in [s, o]$, where $[s, o]$ denotes the entry in the protection matrix for subject s and object o. It is apparent that modelling a reference monitor in this way has certain drawbacks, not least the construction of the deductive database, and may scale rather badly.

We have therefore decided to adopt a different approach and provide an access control authorisation model that implements an order-based access control policy. Additional motivation for this approach is that we believe the three most well-known models, namely

- the Bell-LaPadula model [1],

*Supported by EPSRC Award 98317878

- the protection matrix model [11], and
- the recent role-based access control model [21]

each have drawbacks. The protection matrix and role-based access control models are "policy neutral" - any policy can, in principal, be implemented on systems which are based on these model. However, this generality means that the security properties of such models are difficult to establish. In particular, verifying that a given security policy is being implemented correctly by a reference monitor is difficult. The Bell-LaPadula model, however, has explicit security properties but implements an information flow policy [16] which has proved too restrictive for many commercial systems.

Our hierarchicial secure system model is a generalisation of the Bell-LaPadula model based on an organisational hierarchy and implements policies based on the notion of seniority contained in that hierarchy. The preceding remark suggests that we intend this model to be sufficiently flexible to be used in commercial systems. The hierarchical secure system model is of greater value than the protection and role-based access control models in that it is possible to define a *secure state* of the system. We hope to prove that a secure system can be constructed.

The remainder of the paper is organised as follows. Section 2 is a brief account of existing access control models, as the hierarchical secure system model draws on concepts and notation from them. Section 3 introduces some theoretical results about partially ordered sets. Section 4 introduces the hierarchical secure system model. In the Conclusion we discuss the numerous opportunities for further research.

2 Access control models

Most approaches to access control modelling incorporate the following features.

- A set of *objects*, O. Objects are passive entities in the model and represent files and directories, for example.
- A set of *subjects*, S. Subjects are active entities and represent (processes and programs executing on behalf of) users.
- A set of *access modes*, A. Access to an objects may be granted to a subject in one or more of the access modes. Typical examples of access modes are *read*, *write* and *execute*, usually abbreviated to r, w and x, respectively.
- A set of *requests*, Q and a set of *decisions*, D. Informally, requests are procedure calls to the (access control) system or reference monitor, which result in decisions (system responses) and changes in the state of the system. For example a subject could issue a request
  - to access an object in a particular mode,
  - to give access to another subject to an object in a particular mode.

  Examples of decisions could be "yes", "no" and "?".
- A set of *states*, T. A state, t, is a pair (V, M) where V specifies what is currently authorised, and M what is potentially authorised. The actual characteristics of M will depend on the particular access control model under consideration. V is typically a set of triples (s, o, a).

  To make clear the distinction between what V and M, consider the following two hypothetical requests
  $$\mathtt{get}(s, o, r) \quad \text{and} \quad \mathtt{give}(s_o, s_f, o, r).$$
  The first request asks that subject s be allowed to access mode r to object o. If the decision (system response) is "yes", then (s, o, r) is added to V, while M is unchanged. The second request asks that subject s_o (owner) confers on s_f (friend) the right to access object o in access mode r. If the decision is "yes", then (s_f, o, r) is added to M, while V is unchanged.

- A state transition relation, $\Delta \subseteq T \times Q \times D \times T$, where $\delta = (t, q, d.t') \in \Delta$ if, and only if, when in state t a request q results in the decision d and state t'. The relation Δ captures the (possibly non-deterministic) rules of operation of the system.
- Given a countably infinite (index) set, $I = \{1, 2, \ldots\}$, which is used to model the flow of time, and an initial state, t_0, an *evolution*,

  $$\sigma = (q^i, d^i, t^i) \stackrel{\text{def}}{=} (q_1 \ldots q_i, d_1 \ldots d_i, t_0 t_1 \ldots t_i)$$

  of the system models a possible development of the system at time $i > 0$. An evolution is defined recursively as follows:

  – if $(t_0, r_1, d_1, t_1) \in \Delta$, then

  $$\sigma_1 = (r_1, d_1, t_0 t_1) \text{ is an evolution;}$$

  – if $(t_{i-1}, r_i, d_i, t_i) \in \Delta$ and $\sigma_{i-1} = (r^{i-1}, d^{i-1}, t^{i-1})$ is an evolution, then

  $$\sigma_i = (r^{i-1} r_i, d^{i-1} d_i, t^{i-1} t_i) \text{ is an evolution.}$$

  An evolution captures a particular history of the system at time i as strings of requests, decisions and states[1].

  The system is modelled by the set of evolutions, $\Sigma(\Delta, t_0, Q, D)$. (That is, a system is dependent on the state transition relation, the initial state, the set of requests and the set of decisions.) When the context is clear we will simply write Σ to denote the system.

2.1 The protection matrix model

We first consider the protection matrix model as it forms part of the Bell-LaPadula model. A protection matrix, M, has columns indexed by objects and rows indexed by subjects. Each entry in M is a subset of A, the set of access modes. $[s, o]$ denotes the entry in M for object o and subject s.

A request by subject s to access object o in mode a (which we will denote $\texttt{get}(o, s, a)$) is granted provided $a \in [s, o]$. Typically, protection matrices are sparse and reference monitors which are based on this model often implement M as *access control lists* or *capability lists* which correspond to columns and rows of M, respectively [11].

In the context of the protection matrix model, a state, t, is a pair (V, M), where M is the protection matrix, and V is the set of currently authorised pairs. (V is taken from acti<u>v</u>e.)

It is recognised that the protection matrix model has broad expressive power but that the "security problem" is, in general, undecidable [10].

The security problem is the determination of whether or not a given subject can acquire a particular access mode to a given object. Equivalently, is it possible for a given triple, (s, o, r), to enter V? In a practical sense, the security problem is "given a specification of security (an *access control policy*) does the implementation coincide with policy?". It is sometimes said that the protection model has "weak security properties".

The typed access matrix model [17] imposed strict typing on objects, the types forming a strict order. It was shown that this model has stronger security properties than the protection matrix model, but that the security problem is still **NP**-hard in its decidable cases.

2.2 The Bell-LaPadula model

The Bell-LaPadula model includes the following additional elements.

[1] An evolution is referred to as an *appearance* in the Bell-LaPadula model.

- A set of *security classifications* or *security labels*, C, which is a total order. That is, for all $c_1, c_2 \in C$ either $c_1 \leqslant c_2$ or $c_1 \geqslant c_2$. The most common set of security labels, used in military systems, is
$$\texttt{unclassified} < \texttt{classified} < \texttt{secret} < \texttt{top secret}.$$
- A set, K, of (*needs-to-know*) *categories.* These usually represent different projects or domains.
- A security lattice, $L \subseteq C \times \mathcal{P}(K)$, where $(c_1, K_1) \leqslant (c_2, K_2)$ if, and only if, $c_1 \leqslant c_2$ and $K_1 \subseteq K_2$.
- A set of *security clearance functions*, Λ, where for all $\lambda \in \Lambda$, $\lambda : O \cup S \to L$. The essential idea of the Bell-LaPadula model is that every subject and object is assigned a security clearance by λ.
- A set of $n \times m$ protection matrices, $\mathcal{M}$, where $n = |O|$ and $m = |S|$.
- A set of *states*, $T \subseteq \mathcal{P}(O \times S \times A) \times \mathcal{M} \times \Lambda$. A state $t = (V, M, \lambda) \in T$, consists of $V \subseteq O \times S \times A$ which models the currently authorised access right triples, and M and λ, the current protection matrix and security clearance function, respectively, model the potential authorisations.

When particular access modes are introduced into the Bell-LaPadula model they are divided into two generic groups, A_r, read-type access modes, and A_w, write-type access modes. For convenience, given a state $t = (V, M, \lambda)$ we will denote the set of *read triples* by V_r, and the set of *write triples* by V_w. Formally, $V_r = \{(s, o, a) \in V : a \in A_r\}$ and $V_w = \{(s, o, a) \in V : a \in A_w\}$, where $V_r \cup V_w = V$. It is not necessarily the case that $V_r \cap V_w = \emptyset$, since some access modes may be both read and write access modes. For example, in the Bell-LaPadula model, the write access mode actually meant simultaneous read and write access, while the read and append access modes were read-only and write-only access modes, respectively.

This actually makes the enforcement of a confidentiality or information flow policy rather hard. Read-type access modes present no problem on their own (see the simple security property below). However, indirect breaches of confidentiality can arise if a copy of an object can be written to a less secure object by a subject which has read access to the former and write access to the latter (see the *-property).

Simple security property A state $t = (V, M, \lambda)$ satisfies the *simple security property* if, and only if, for all $v = (s, o, a) \in V_r$, $\lambda(s) \geqslant \lambda(o)$. In particular, if a get request, $\texttt{get}(s, o, a)$ where $a \in A_r$, is granted then $\lambda(s) \geqslant \lambda(o)$. (Otherwise, the resulting state will not satisfy the simple security property.)

The simple security property expresses the requirement that no user should be able to read an object that has a higher security clearance than the user.

***-property** In order to more readily describe the *-property, define
$$V(s) = \{o \in O : (s, o, a) \in V\}.$$
That is, $V(s)$ is the set of objects in a state V to which a particular subject s has at least one access mode.

A state $t = (V, M, \lambda)$ satisfies the **-property* if, and only if,
$$\text{for all } s \in S,\ o_1 \in V_w(s),\ o_2 \in V_r(s),\ \lambda(o_1) \geqslant \lambda(o_2),$$
That is, a state satisfies the *-property if, and only if, for all subjects, the objects a subject can write to are at least as secure as the objects that subject can read.

The *-property expresses the requirement that no subject should be able to write (or "leak" information) to an object which is less secure than the subject. In particular, if a get request,

$\texttt{get}(s, o, a)$ where $a \in A_w$, is granted then $\lambda(s) \leqslant \lambda(o)$. The examples usually quoted for this rather counter-intuitive condition are

- to prevent Trojan horse software copying `top secret` material to an `unclassified` file, say;
- to prevent the writing of `top secret` material to an `unclassified` printer, for example.

Figure 1 shows a schematic representation of the evaluation of a `get` request for read access in the Bell-LaPadula model. (In fact, this evaluation process is more complicated, since the addition of any read triple may conflict with the *-property because of write triples in the set of currently authorised triples, V [1].)

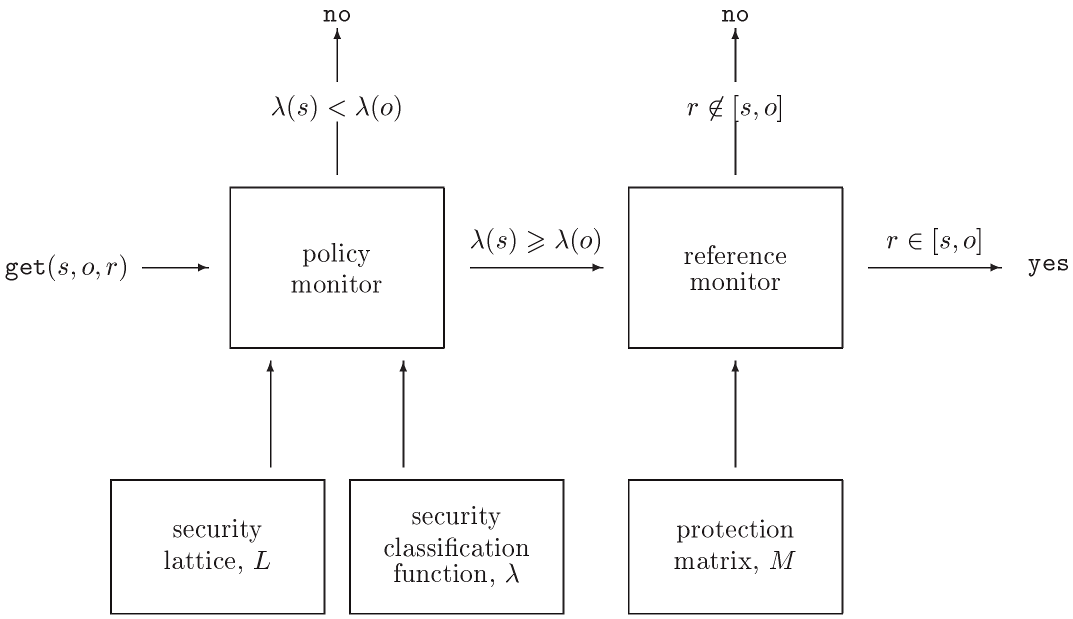


Figure 1: Evaluating a read request in the Bell-LaPadula model

Secure systems A state $t = (V, M, \lambda)$ is *secure* if V satisfies the simple security property and the *-property. An evolution $(q_1 \dots q_i, d_1 \dots d_i, t_0 t_1 \dots t_i)$ is secure if t_j is secure, for all j, $1 \leqslant j \leqslant i$. A system is secure if every evolution is secure.

The Bell-LaPadula model provides a rigorous definition of system security and, furthermore, demonstrates how to construct a secure system. In other words, the Bell-LaPadula model has very strong security properties.

Remark 2.1 *It is interesting to note that in the Bell-LaPadula paper the entries in the protection matrix are not required to satisfy the information flow policy. Specifically it is theoretically possible for there to exist $s \in S$, $o \in O$, $a \in A_r$ such that $\lambda(s) < \lambda(o)$ and $a \in [s, o]$. (This does not represent a breach of policy since the policy monitor will deny the request $\texttt{get}(s, o, a)$.)*

2.3 The role-based access control model

The formal development of this model began in the mid-90s [8, 9, 13, 20], although many of the concepts had been incorporated into operating systems much earlier.

The simplest formulation of the role-based access control model includes the following features.

- A set of *roles*, R
- A set of *users*, U
- The set of access modes is replaced by a set of *permissions*, A. Permissions are typically treated as "uninterpreted symbols" in role-based access control models. Our interpretation of the literature is that so-called primitive permissions are pairs (o, a) where $o \in O$ and $a \in A$; and that permissions in general can be thought of as sets of primitive permissions.
- A *user-role assignment relation*, $UA \subseteq U \times R$
- A *permission-role assignment relation*, $PA \subseteq A \times R$
- A *role hierarchy*, $RH \subseteq R \times R$, which can also be thought of as a poset $\langle R, \leqslant\rangle$. Specifically, for $r_1, r_2 \in R$, $r_1 \leqslant r_2$, if, and only if, $(r_1, r_2) \in RH$. Permission assignments to roles are inherited by more senior roles in the hierarchy. Users are *implicitly* assigned to roles which are junior to any roles they are (*explicitly*) assigned to in UA.
- A *session* is analogous to a subject in traditional models. A session is uniquely identified with a user and is some subset of the roles to which the user is implicitly assigned. A permission, p, is granted to a session, s if there exists a role $r \in s$ such that $(p, r) \in PA$.

There is, in addition, an administrative model, ARBAC97 [19], which specifies how the role hierarchy, user-role assignment relation and permission-assignment relation should be managed. The administrative role hierarchy is a set of administrative roles, AR, which is disjoint from R. The user-role assignment and permission-role assignment relations are extended to include AR. There is also a set of administrative permissions which can only be assigned to administrative roles.

Assignments and revocations of user-role assignments are controlled by a `can-assign` relation. An element of this relation consists of an administrative role identifier, ar, a constraint, c, and a set of roles, *range*. The administrative role, ar, can assign any role in *range* to any user which satisfies c. Similar relations exist for revoking a user-role assignment, creating a permission-role assignment and revoking a permission-role assignment.

For the simplest role-based access control model, a state is a 4-tuple (V, RH, UA, PA), where V models the currently authorised permissions, and RH, UA and PA model the potentially authorised permissions. An element of V is a triple (s, o, a), where s is a session which contains a role r to which the permission (o, a) is assigned. However, the inclusion of ARBAC97 in the role-based access control model, for example, implies a state should also include the relations `can-assign`, `can-revoke`, `can-assignp` (determining permission-role assignments), `can-revokep` (determining permission-role revocations).

It is not clear how difficult the security problem is for this model[2]. It is left as an exercise for the interested reader! We note in [3] that inappropriate entries or the omission of certain entries in the `can-assign` relation can lead to user-role assignments which would clearly compromise security. It is also possible within the framework that has been defined that two entries in the relation may be contradictory. Specifically, ar may be authorised to assign u to a role $r \in range$ by one tuple in `can-assign` and may not be authorised by another tuple. We discuss an example of a role-based access control system in detail in Section 5.1.

In short, despite the intuitive appeal of the role hierarchy and the reduction in the administration of permissions, the value and behaviour of the administrative model is far less clear, and hence modelling the possible evolutions of a role-based access control system appears very difficult.

[2]We conjecture that it is at least as hard as the security problem for the protection matrix model, not least because a simulation of the protection matrix model using a role-based model has been constructed by Sandhu *et al* [14].

3 Completions of a poset

We now outline a theoretical result in the theory of partially ordered sets (posets) which arose out of modelling conflict of interest policies in access control [4, 5]. A particular case of this result provides a natural way of generating the ordering in the security lattice, and was one of the motivations for this work.

Definition 3.1 *A pair* $\langle X, \leqslant \rangle$ *is a* partially ordered set *or* poset *if for all* $x, y, z \in X$*,*

- $x \leqslant x$*,*
- $x \leqslant y$ *and* $y \leqslant x$ *implies* $x = y$*,*
- $x \leqslant y$ *and* $y \leqslant z$ *implies* $x \leqslant z$*.*

In other words $\leqslant$ is a binary relation on X which is reflexive, anti-symmetric and transitive, respectively. We will write

- $x < y$ if, and only if, $x \leqslant y$ and $x \neq y$; and
- $x \parallel y$ if, and only if, $x \nleqslant y$ and $x \ngeqslant y$.

In the remainder of this section, we will write X to mean the pair $\langle X, \leqslant \rangle$.

Definition 3.2 *Given a poset* X*,* $Y \subseteq X$ *is a* chain *if for all* $y_1, y_2 \in Y$ *either* $y_1 \leqslant y_2$ *or* $y_2 \leqslant y_1$*.* Y *is an* antichain *if for all* $y_1, y_2 \in Y, y_1 \parallel y_2$*. We denote the set of antichains by* $\mathcal{A}(X)$*.*

Definition 3.3 *Given a poset* X *and* $Y \subseteq X$*, we say* $y \in Y$ *is a* minimal *element if for all* $y' \in Y$*,* $y' \leqslant y$ *implies* $y = y'$*. Similarly,* $y \in Y$ *is a* maximal *element if for all* $y' \in Y$*,* $y \leqslant y'$ *implies* $y = y'$*. We denote the set of minimal elements in* Y *by* $\underline{Y}$*; and the set of maximal elements in* Y *by* $\overline{Y}$*.*

Lemma 3.1 *For all* $Y \subseteq X$*,* $y \in Y$*,*

$$\underline{Y} \subseteq Y, \tag{1}$$

$$\text{there exists } y' \in \underline{Y} \text{ such that } y' \leqslant y, \tag{2}$$

$$\underline{Y} \in \mathcal{A}(X), \tag{3}$$

$$\underline{Y} \text{ is unique.} \tag{4}$$

Proof The proof follows immediately from Definition 3.3. ∎

Definition 3.4 *Given a poset* X *and* $x, y, \in X$*, we say* y covers x*, denoted* $x \lessdot y$*, if* $x < y$ *and* $x \leqslant z < y$ *implies* $x = z$*.*

Definition 3.5 *Let* X *be a poset, and let* $Y \subseteq X$*.*

- *An element* $x \in X$ *is an* upper bound *for* Y *if, for all* $y \in Y$*,* $y \leqslant x$*.*
- *An element* $x \in X$ *is a* least upper bound *or* supremum *for* Y*, denoted* $\sup Y$*, if* x *is an upper bound of* Y *and, for all* $y \in Y$*,* $z \in X$*,* $y \leqslant z$ *implies* $x \leqslant z$*. In other words,* x *is the smallest of the upper bounds of* Y*.*
- *An element* $x \in X$ *is a* lower bound *for* Y *if, for all* $y \in Y$*,* $x \leqslant y$*.*
- *An element* $x \in X$ *is a* greatest lower bound *or* infimum *for* Y*, denoted* $\inf Y$*, if* x *is a lower bound of* Y *and, for all* $y \in Y$*,* $z \in X$*,* $z \leqslant y$ *implies* $z \leqslant x$*. In other words,* x *is the greatest of the lower bounds of* Y*.*

Definition 3.6 *A poset,* X*, is a* lattice *if, and only if, for all* $x, y \in X$ *both* $\inf\{x, y\}$ *and* $\sup\{x, y\}$ *exist in* X*. If for all* $Y \subseteq X$*,* $\sup Y$ *and* $\inf Y$ *exist (in* X*), then* X *is called a* complete *lattice.*

If L is a lattice, then

$$\inf\{x,y\} \quad x \wedge y \quad \textit{meet} \text{ of } x \text{ and } y$$

are equivalent, as are

$$\sup\{x,y\} \quad x \vee y \quad \textit{join} \text{ of } x \text{ and } y.$$

We write $\langle L, \vee, \wedge\rangle$ to mean that the set L is a lattice with the operations $\vee$ and $\wedge$. Indeed a lattice can be defined as a purely algebraic structure in terms of these operations [2].

Definition 3.7 *Given two posets $\langle X_1, \leqslant_1\rangle$ and $\langle X_2, \leqslant_2\rangle$, a function $\phi : X_1 \to X_2$ is an* order-embedding *if for all $x, y \in X_1$, $x \leqslant_1 y$ implies $\phi(x) \leqslant_2 \phi(y)$. If X_2 is a complete lattice, we say X_2 is a* completion *of X_1 (via the order-embedding ϕ).*

Theorem 3.1 *Given a poset $\langle X, \leqslant\rangle$, define $\mathcal{A}(X) = \{\alpha \subseteq X : \alpha \text{ is an antichain}\}$ and for all $\alpha, \beta \in \mathcal{A}(X)$,*

$$\alpha \preccurlyeq \beta \text{ if, and only if, for all } a \in \alpha, \text{ there exists } b \in \beta \text{ such that } a \leqslant b.$$

Then $\langle \mathcal{A}(X), \preccurlyeq\rangle$ is a lattice. Moreover, $\langle \mathcal{A}(X), \preccurlyeq\rangle$ is a completion of X via the mapping $x \mapsto \{x\}$.

Proof The proof is given in [4]. ■

It can be shown that for the poset $C \cup K$, where C is the set of security labels and K is the set of categories, $\preccurlyeq$ corresponds to the ordering on the security lattice $C \times \mathcal{P}(K)$. Figure 2a, shows the set $C \cup K$, where `unclassified`, `classified`, `secret`, and `top secret` have been abbreviated to `u`, `c`, `s` and `ts`, respectively; and C_1 and C_2 are needs-to-know categories. The completion of $C \cup K$ using antichains, $\mathcal{A}(C \cup K)$ is shown in Figure 2b. (Set delimiters ",", "{", "}") have been omitted in the nodes of $\mathcal{A}(P)$.)

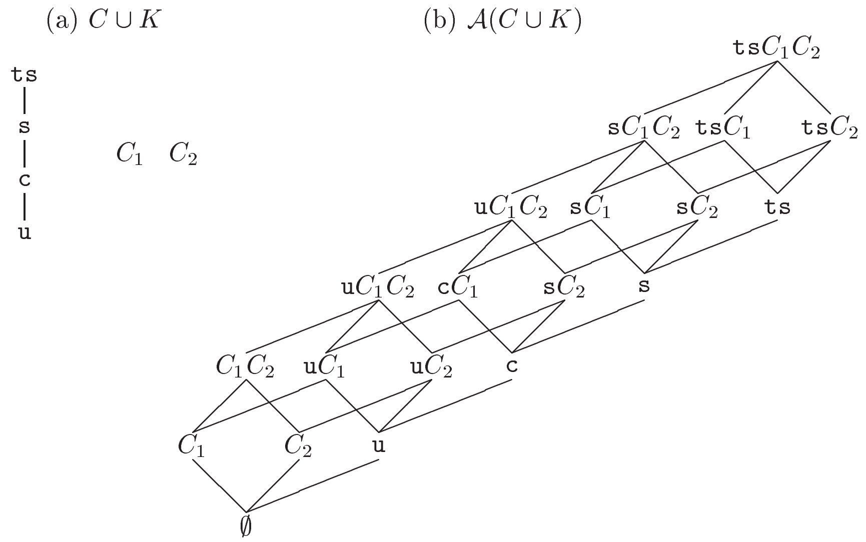


Figure 2: A Bell-LaPadula "hierarchy" and its completion

Theorem 3.2 *Given a poset X and $Y \subseteq X$, define*

$$Y^{\underline{u}} = \{x \in X : \textit{for all } y \in Y,\ y \lessdot x\}$$
$$Y^{\overline{l}} = \{x \in X : \textit{for all } y \in Y,\ x \lessdot y\}$$

Then $\langle DM_A(X), \preccurlyeq\rangle$ is a lattice, where

$$DM_A(X) = \{Y \subseteq X : Y^{\underline{u}\overline{l}} = Y\}.$$

Furthermore, $DM_A(X)$ is a completion of X via the mapping $x \mapsto \{x\}$.

Proof The proof is given in [4]. ■

It can easily be proved that $DM_A(X) \subseteq \mathcal{A}(X)$ [4].

4 The hierarchical secure system model

We want to define a model which couples strong security properties with a reasonable and realistic view of the requirements of security in a commercial system. To achieve this we make use of the observation that most organisations have a *supervision hierarchy* based on superiority of position within the organisation. A supervision hierarchy is described in [12] as

> *"... a hierarchy of named positions ... each position has one or more roles ... these roles can be supervisory (administrative), or functional or both."*

It seems reasonable, therefore, to assume that access to objects can be identified with positions in a supervision hierarchy. Using the obvious parallel with security labels in the Bell-LaPadula model, we informally have the idea of a security policy in which access to an object is granted to a subject only if the position of the subject is greater than that of the object.

We assume, unlike the Bell-LaPadula model, that objects and subjects can be associated with a set of positions, thus giving our model greater flexibility. By regarding the supervision hierarchy as a partial order on a set of positions and using the results of Section 3, we show how to extend the ordering of the supervision hierarchy to an ordering on sets of positions in a completion of the supervision hierarchy.

We also take the view that any non-policy-specific access control model can be used as the reference monitor in the Bell-LaPadula model. Specifically, we could use either a protection matrix or the "machinery" of the role-based access control model. The choice would be determined by the application domain.

We now formally define the characteristics of hierarchical secure system model, following the spirit of the seminal paper of Bell and LaPadula [1].

4.1 The theoretical framework

We introduce the following definitions.

- A partially ordered set of *positions*, $\langle P, \leqslant \rangle$. The covering relation defines the position hierarchy and corresponds to the supervision hierarchy mentioned above.

  By Theorem 3.1, there exist two completions of P using antichains (that is two complete lattices, which preserve the ordering of P), namely $\mathcal{A}(P)$ and $DM_{\mathcal{A}}(P)$.

- A *reference monitor*, M, which may be based on the protection matrix model, the role-based access control model, or any other policy-neutral access control model. The interpretation of subjects will be determined by the reference monitor chosen. For example, in the case of a role-based access control reference monitor, subjects will be interpreted as sessions.

- A *seniority function*, ϕ. The purpose of ϕ is to associate all entities in the system with a level of seniority within the organisational framework. This is clearly analogous to the security clearance function, λ, in the Bell-LaPadula model.

  The domain of ϕ will depend on the reference monitor. For example, using a role-based access control reference monitor, we would define ϕ for all objects, roles and users.

  The range of ϕ is $\mathcal{A}(P)$. In other words, the seniority of an entity is a set of positions which are pairwise incomparable (with respect to the position hierarchy). Typically this set will be a singleton and hence will coincide with the usage of λ in the Bell-LaPadula model. However, the fact that $\mathcal{A}(P)$ is a complete lattice means that a more senior antichain of positions can be found for any antichain of positions, and hence that this definition of ϕ is both legitimate and provides more flexibility than the definition of λ. We note the following possibilities.

- $\phi(o) = \emptyset$ - that is, o is not protected by the policy, as every subject will be senior to o. This corresponds intuitively to public documents.
- $\phi(s) = \emptyset$ - that is, s has no access privileges except to public documents. This could model, for example, a guest user.

- An *order-based policy* (or *seniority policy* or simply *policy*), Π, is a set of (policy) statements, $\{\pi_1, \ldots, \pi_n\}$, where each π_i is an inequality relating the seniority of entities within the system. Informally, an order-based policy is one in which the specification of the policy can be stated in terms of the ordering on the position hierarchy. Some examples of statements are

  - $\pi_1 = \phi(s) \succcurlyeq \phi(o)$, which is analogous to the simple security property;
  - $\pi_2 = \phi(s) \preccurlyeq \phi(o)$, which is analogous to the *-property;
  - $\pi_3 = \phi(r) \preccurlyeq \phi(u)$, which relates the seniority of roles to that of users.

  The policy $\Pi_{BLP} = \{\pi_1, \pi_2\}$ is analogous to the information flow policy of the Bell-LaPadula model.

- A *policy monitor* which is used to enforce an *order-based policy.* In particular, such a monitor could be used to check the simple security property, the *-property, or the Biba integrity policy.

  Our model, therefore, like the Bell-LaPadula model, has a two stage protocol for determining the system response to a request. That is, the request must satisfy the seniority policy and it must be granted by the reference monitor.

- A set of *requests* and a set of *decisions*, D.

- A set of *states*, T. A *state* is a triple (ϕ, M, V) where M models the potential authorisations of the reference monitor, and V models the set of subjects which the current authorisations. For example, using a role-based access control reference monitor, $M = RH \cup UA \cup PA$, and V is a set of triples (as with a protection matrix) comprising a session and a capability.

  We define a *state transition relation* $\Delta \subseteq T \times Q \times D \times T$, where $\delta \in \Delta$ if, and only if, when in state $t = (t, q, d, t')$ a request q results in the decision d and state t'. The relation Δ captures the (possibly non-deterministic) rules of operation of the system.

Hierarchical secure systems A state is *secure with respect to* Π provided all triples $v \in V$ satisfy all $\pi \in \Pi$. An evolution (q^i, d^i, t^i) is *secure with respect to* Π if t_j is secure for all j such that $1 \leqslant j \leqslant i$. A system Σ is *secure with respect to* Π if all evolutions of the system are secure.

4.2 Summary

We have the following characteristic features of our model.

[1] Hierarchical secure system model is extension of Bell-LaPadula model

[2] Strict order and categories replaced with more general hierarchy reflecting organisational characteristics

[3] Security clearance function replaced with more general seniority function

[4] Two stage checking of access requests

These features provide the following advantges over existing models.

- [2] and [3] suggest greater flexibility (than the Bell-LaPadula model)
- [1] and [4] guarantee strong security properties

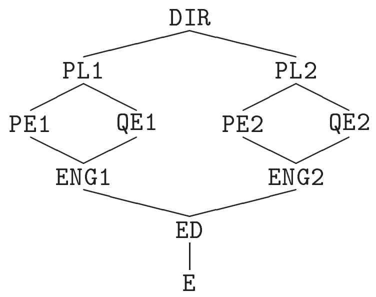
5 Examples and special cases

We now present an example of a hierarchical secure system. It is a very simple example based on one of the most commonly used examples in the role-based access control literature [19]. We use a role-based access control reference monitor, and initially consider a system which enforces no policy. We show that this system has features which we believe are an improvement on existing role-based access control models. We then introduce a simple security policy and demonstrate that this results in a useful administrative model for role-based access control.

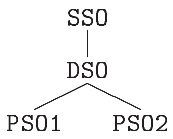
5.1 The role-based access control example

Role hierarchies are represented by a *role graph*, the transitive reduction of the graph of the relation RH. (In other words, a role graph is a Hasse diagram of the poset $\langle R, \leqslant \rangle$.) An example of a role graph is shown in Figure 3. This example is taken from the seminal paper on role-based access control models [20].

(a) Role hierarchy



(b) Administrative role hierarchy



DIR	Director
PL	Project Leader
PE	Production Engineer
QE	Quality Engineer
ENG	Engineer
ED	Engineering Department
E	Employee
SSO	Senior Security Officer
DSO	Departmental Security Officer
PSO	Project Security Officer

Figure 3: RBAC96 Hierarchies

In Table 1 we show some typical user-role assignments. Note for each user, u, $R(u) \in \mathcal{A}(R)$ for the poset R in Figure 3a. (Clearly we have made some assumptions in this assignment of roles to users. For example, we have assumed that the Director would most likely be assigned the Senior Security Officer role, and that the Project Security Officer roles would be assigned to Project Leaders. We note, however, that there is no constraint within the example, that prevents `dave`, for example, being assigned the Senior Security Officer role.)

In Table 2 we show some typical permission-role assignments, possible choices for $R(s)$ and the corresponding permissions for a session, s, where $U(s) = \texttt{bill}$.

An administrative model, ARBAC97, was introduced in [19] to manage the assignment of users and permissions to roles, and changes to the role hierarchy. Table 3 shows examples of two of the relations introduced in ARBAC97 with reference to Figure 3. Namely, the `can-assign` and `can-revoke` relations which manage the assignment of users to roles.

If we consider the `can-assign` relation we see that `PSO1` can assign a user to roles `ENG1`, `PE1` and `QE1` (but not `PL1` because the upper end-point is not included in the range) provided the user

u	$R(u)$	$\downarrow R(u)$
anne	QE1, QE2	QE1, ENG1, ED, E, QE2, E2
bill	PL1, PSO1	PL1, PE1, QE1, ENG1, ED, E, PSO1
claire	DIR, SSO	DIR, PL1,..., ED, E, PL2,..., ENG2, SSO, DSO, PSO1, PSO2
dave	ENG1	ENG1, ED, E
emma	PE1, QE2	PE1, ENG1, ED, E, QE2, E2

Table 1: User-role assignments in RBAC

r	$P(r)$	$\downarrow P(r)$
ENG1	p_1	p_1
PE1	p_2	p_1, p_2
QE1	p_3	p_1, p_3
PL1	p_4	p_1, p_2, p_3, p_4

$R(s)$	$\downarrow R(s)$	$P(s)$
ENG1	ENG1	p_1
PE1	PE1, ENG1	p_1, p_2
QE1	QE1, ENG1	p_1, p_3
PE1, QE1	PE1, QE1, ENG1	p_1, p_2, p_3
PL1	PL1, PE1, QE1, ENG1	p_1, p_2, p_3, p_4

Table 2: Permissions, roles and sessions

satisfies the constraint ED; in other words, provided the user is assigned a role at least as senior as ED. DSO can assign a user in ED to role PL2 provided the user is not already assigned to any roles at least as senior as PL1.

Table 4 shows examples of users and whether each user satisfies the constraints of Table 3. (We use T to denote that a constraint is satisfied, and F otherwise.)

Table 5 shows some potential user-role assignments. r denotes a role which can be assigned to u, ar denotes the set of administrative roles that can assign r to u, and $R(u)'$ denotes the resulting explicit role assignments to u. We make the following observations about the can-assign relation.

- Since the constraints can be arbitrarily complex conjunctions and disjunctions of constraints, we conjecture that the problem of deciding whether a given user can be assigned to a given role is at least **NP**-hard and may be undecidable.

- dave can be assigned the role PL1 for example, which (presumably) permits considerably greater privileges than he previously had.

- bill cannot be assigned to PL2 because of the constraint in row 3 of Table 3 (which is explicitly evaluated in Table 4), but can be assigned to roles less senior that PL2.

- The only elements of the can-assign relation that are relevant to bill are the second and third rows. That is, there is redundancy in the can-assign relation.

- fred cannot be assigned to any roles because he does not satisfy any constraints.

5.2 The equivalent hierarchical secure system example

We use the example of Section 5.1 as our starting point. We will show that the use of a very simple and intuitively natural position hierarchy leads to greater flexibility in the definition of the role hierarchy.

It has been observed that the "overloading" of the role hierarchy may be inappropriate [12, 22]. The role hierarchy is overloaded in the sense that it serves a dual purpose defining both permission inheritance and role activation.

There have at least two attempts to address this issue resulting in the concept of a *private* role [20] and the extended role-based access control model (ERBAC) [18]. There are no roles senior to a private role so that any permissions assigned are exclusive to that role. In ERBAC

can-assign		
Administrative Role	Constraint	Role Range
PSO1	ED	[ENG1, PL1)
PSO2	ED	[ENG2, PL2)
DSO	ED ∧ $\overline{\text{PL1}}$	[PL2, PL2]
DSO	ED ∧ $\overline{\text{PL2}}$	[PL1, PL1]

can-revoke	
Administrative Role	Role Range
PSO1	[ENG1, PL1)
PSO2	[ENG2, PL2)
DSO	[ED, DIR]

Table 3: URA97 predicates

u	$R(u)$	Constraint				
		ED	$\overline{\text{PL1}}$	$\overline{\text{PL2}}$	ED ∧ $\overline{\text{PL1}}$	ED ∧ $\overline{\text{PL2}}$
anne	{QE1, QE2}	T	T	T	T	T
bill	{PL1}	T	F	T	F	T
dave	{ENG1}	T	T	T	T	T
fred	$\emptyset$	F	T	T	F	F

Table 4: Users and constraint satisfaction in URA97

there are two hierarchies - an *inheritance* hierarchy, IH, and an *activation* hierarchy, AH. The former governs the inheritance of permissions, and the latter the roles to which a user is implicitly assigned. $r_1 <_{IH} r_2$ implies that $r_1 <_{AH} r_2$. In Figure 5(b), PE1′ is an example of a private role; while Figure 5(c) shows the ERBAC hierarchies. A broken line connecting r_1 to r_2 indicates that $(r_1, r_2) \notin IH$ and $(r_1, r_2) \in AH$. We will illustrate that both of these solutions seem rather contrived compared to the natural consequences of using a position hierarchy and role hierarchy.

Figure 4a shows one possibility for the role hierarchy of Figure 3. We have assumed that certain roles, like E, ED and PL have permissions which may plausibly be inherited by other roles. However, we have removed the inheritance between roles such as DIR and PL1. The resulting hierarchy has some similarity to the ERBAC inheritance hierarchy introduced in [18]. This hierarchy was introduced to make a distinction between permission inheritance and role inheritance as it was recognised that permission inheritance should not necessarily be implied by seniority.

However, under the assumption that a user can activate any roles which are less than or equal in seniority to the user, we preserve the role activation hierarchy. Because we have introduced an additional hierarchy, this feature arises naturally in our model. Note that this assumption is equivalent to the following *role activation property*, π_{ra}. That is, for a user, u, to activate a role, r,

$$\phi(u) \succcurlyeq \phi(r)$$

must be satisfied. Such a policy means we can dispense with the UA assignment relation - it has been replaced by ϕ, effectively a user-position assignment relation.

Note that with the current values of ϕ a user who is an Engineer can be assigned to any of the roles PE1, QE1, PE2, QE2, ENG1, ENG2, ED, E. We can restrict these assignments further, by introducing positions (analogous to categories) P1 and P2, for example. Suppose now that $\phi(\texttt{emma}) = \{\text{Engineer}, \texttt{P1}\}$, then emma can only be assigned to the roles PE1, QE1, ENG1 ED1 (assuming analogous changes are made to the role-position assignments). (Alternatively, we can include a user-role assignment relation, UA, either in addition to or instead of π_{ra}.)

5.3 A simple administrative security policy

We now introduce a simple order-based policy requirement. For the assignment of a user, u, to a role, r, by a role r_a, the *user assignment property*, π_{ua}, namely

$$\phi(r_a) \succ \phi(u) \succcurlyeq \phi(r),$$

u	$R(u)$	r	ar	$R(u)'$
anne	{QE1, QE2}	PE1	{PSO1, DSO, SSO}	{QE1, QE2, PE1}
anne	{QE1, QE2}	PL1	{DSO}	{PL1, QE2}
bill	{PL1}	PE2	{PSO2, DSO, SSO}	{PL1, PE2}
dave	{ENG1}	PL1	{DSO}	{PL1}

Table 5: User-role assignments in URA97

must be satisfied. We believe that these are very natural conditions to impose and are a much more accurate reflection of how responsibilities are assigned in an organisation. Specifically, personnel are not usually given the opportunity or responsibility for tasks or roles for which they are not qualified (by seniority). Note that, for example, `dave` cannot now be assigned to the role `PL1`, unlike in the ARBAC97 framework.

The administration of permission-role assignments is even simpler. For the assignment of a permission (o, a) to a role r by a role r_a, we require that the *permission assignment property*, π_{pa}, namely

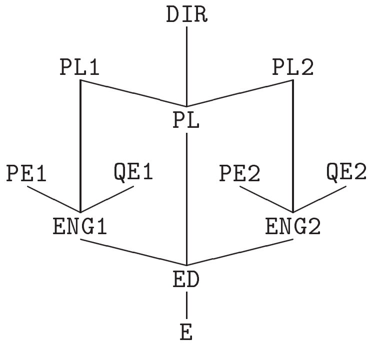
$$\phi(r_a) \succ \phi(r),$$

must be satisfied. We can optionally insist that $\phi(r) \succcurlyeq \phi(o)$, but recalling Remark 2.1, observe that this is not necessary, as the policy monitor will prevent any session exercising this permission without the appropriate seniority.

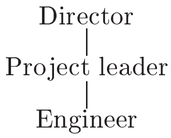
6 Special cases

7 Conclusion

(a) Role hierarchy, R



(b) Position hierarchy, P



(c) User-position associations

u	$\phi(u)$
`anne`	{Engineer}
`bill`	{Project leader}
`claire`	{Director}
`dave`	{Engineer}
`emma`	{Engineer}
`fred`	{Engineer}

(d) Role-position associations

r	$\phi(r)$
`DIR`	{Director}
`PL1`	{Project leader}
`PL`	{Project leader}
`PE1`	{Engineer}
`QE1`	{Engineer}
`ENG1`	{Engineer}
`ED`	{Engineer}
`E`	$\emptyset$

Figure 4: An example of a hierarchical secure system

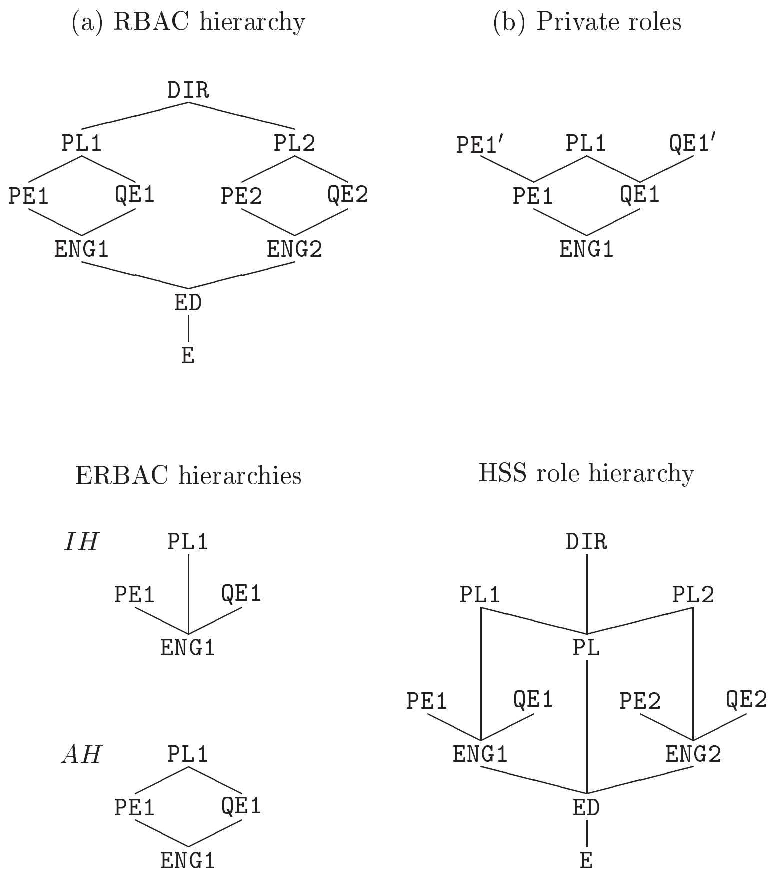


Figure 5: A comparison of role hierarchies

References

[1] D.E. Bell and L. LaPadula. Secure computer systems: Mathematical foundations. Technical Report MTR-2547, Volume I, Mitre Corporation, March 1973.

[2] S. Burris and H.P. Sankappanavar. *A Course in Universal Algebra*. Springer-Verlag, New York, 1981.

[3] J. Crampton and G. Loizou. Role-based access control: A constructive appraisal. In preparation.

[4] J. Crampton and G. Loizou. The completion of a poset in a lattice of antichains. *Submitted*, October 2000. Preliminary version available as Technical Report BBKCS-0001.

[5] J. Crampton and G. Loizou. On the structural complexity of conflict of interest policies. To be submitted, November 2000.

[6] J. Crampton, G. Loizou, and G. O'Shea. Evaluating access control. Technical Report BBKCS-9905, Birkbeck College, University of London, 1999.

[7] J. Crampton, G. Loizou, and G. O'Shea. A logic of access control. *The Computer Journal*, 2000. Accepted for publication.

[8] D.F. Ferraiolo and D.R. Kuhn. Role-based access control. In *Proceedings of the 15th NIST-NSA National Computer Security Conference*, pages 554–563, Baltimore, Maryland, 1992.

[9] D.F. Ferriaolo, J.A. Cugini, and D.R. Kuhn. Role-based access control (RBAC): Features and motivations. In *Proceedings of 11th Annual Computer Security Applications Conference*, pages 241–248, New Orleans, Louisiana, December 1995.

[10] M.A. Harrison, W.L. Ruzzo, and J.D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, August 1976.

[11] B.W. Lampson. Protection. *ACM Operating Systems Review*, 8:437–443, 1974.

[12] J. Moffett and E.C. Lupu. The uses of role hierarchies in access control. In *Proceedings of Fourth ACM Workshop on Role-Based Access Control*, pages 153–160, Fairfax, Virginia, October 1999.

[13] M. Nyanchama and S. Osborn. The role graph model. In *Proceedings of First ACM Workshop on Role-Based Access Control*, pages II25–II31, Gaithersburg, Maryland, October 1995.

[14] 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*, 3(2), May 2000.

[15] G. O'Shea. *Access Control in Operating Systems*. PhD thesis, Birkbeck College, University of London, July 1997.

[16] R.S. Sandhu. Lattice-based enforcement of Chinese Walls. *Computers & Security*, 11(8):753–763, December 1992.

[17] R.S. Sandhu. The typed access matrix model. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 122–136. IEEE, May 1992.

[18] R.S. Sandhu. Role activation hierarchies. In *Proceedings of Third ACM Workshop on Role-Based Access Control*, pages 33–40, Fairfax, Virginia, October 1998.

[19] R.S. Sandhu, V. Bhamidipati, E.J. Coyne, S. Ganta, and C.E. Youman. The ARBAC97 model for role-based administration of roles: Preliminary description and outline. In *Proceedings of Second ACM Workshop on Role-Based Access Control*, pages 41–49, Fairfax, Virginia, November 1997.

[20] R.S. Sandhu, E.J. Coyne, H. Feinstein, and C.E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[21] R.S. Sandhu, D.F. Ferraiolo, and D.R. Kuhn. The NIST model for role-based access control: Towards a unified standard. In *Proceedings of the 5th ACM Workshop on Role-Based Access Control*, Phoenix, Arizona, USA, 2000. http://www.acm.org/sigsac/nist.pdf.

[22] M. Sloman. Policy driven management for distributed systems. *Journal of Network and Systems Management*, 2(4):333–360, 1994.