On Permissions, Inheritance and Role Hierarchies

Jason Crampton

Information Security Group
Royal Holloway, University of London
Egham, TW20 0EX, United Kingdom

jason.crampton@rhul.ac.uk

ABSTRACT

Role-based access control and role hierarchies have generated considerable research activity in recent years. In many role-based models the role hierarchy partially determines which roles and permissions are available to users via various inheritance mechanisms. In this paper, we consider the nature of permissions more closely than is customary in the literature and propose a particular structure for permissions. We then introduce a role-based access control model that contains a novel approach to permission inheritance and illustrate how this model can be used to derive a role-based model with multi-level secure properties. We also consider the issue of redundant and consistent permission-role assignments and describe how such assignments can be avoided.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; I.6.0 [**Computing Methodologies**]: Simulation and Modeling

General Terms

Security, Theory

Keywords

role-based access control, mandatory access control, permissions, inheritance

1. INTRODUCTION

Role-based access control is now widely accepted as an alternative to techniques based on the protection matrix such as access control lists. The basic idea of associating a set of privileges or permissions with a named role and assigning that role to users is well established and is deployed in several commercial computer systems and applications.

CCS'03, October 27–31, 2003, Washington, DC, USA.

The concept of a (role) hierarchy is central to many theoretical role-based access control models [6, 12, 16]. It is customary to use the hierarchy to aggregate permissions; that is, a role is assumed to inherit the permissions assigned to roles below it in the hierarchy. In addition, the role hierarchy also determines the roles that are available to a user; that is, a user assigned to a particular role can also activate any subordinate roles in the hierarchy.

It has been observed that these assumptions have a number of inconvenient consequences [7, 11]. Most importantly, senior roles have access to all permissions assigned to junior roles. This may be inappropriate within many organizations, where senior managers are neither competent nor experienced enough to undertake the activities of more junior positions [7]. Moreover, it becomes impossible to define separation of duty requirements on roles that have a common senior role (unless of course no user is assigned to that senior role) [8]. Finally, the inheritance characteristics of the role hierarchy make it awkward to implement multi-level secure systems using role-based methods. Specifically, all inheritance of permissions is upward within a role hierarchy, unlike write permissions in the Bell-LaPadula model [1].

In an effort to address some of these consequences we consider the effect of using a different model for the aggregation of permissions within the hierarchy. In particular, we assume that permissions can be inherited in one of three ways within the hierarchy: by more senior roles, by less senior roles and by no other roles. The motivation for this approach is supplied by a consideration of certain correlations between the Bell-LaPadula model and role-based access control models. Firstly, we make an explicit connection between the security level of a subject in the former with the user-assignment relation in the latter. Secondly, we note that the acquisition of access rights in the Bell-LaPadula model is governed by the security level of the user, not by permission inheritance. Finally, we observe that the *-property in the Bell-LaPadula model can be considered to be a form of separation of duty which is supported by the use of constraints in role-based models.

We find that our model offers certain advantages over existing role-based models: it simplifies the evaluation of requests by users to exercise permissions; it does not preclude the use of separation of duty constraints; and it provides a more natural implementation of multi-level secure features than has previously been possible using role-based models.

The main contributions of this paper are: to make explicit and strengthen the similarities between role-based and mandatory access control, and to propose a new mechanism

for permission inheritance within a role hierarchy. In doing so, we elucidate the relationships between sessions, users, roles and permissions in role-based models, and construct a simple role-based model for multi-level secure systems.

In the next section we recall the basic features of role-based access control. In Section 3 we describe our approach and in the following section we illustrate how our model can be used to implement multi-level secure systems with the addition of surprisingly few constraints to the basic model. Finally, we summarize the contributions of the paper and discuss opportunities for further research.

2. ROLE-BASED ACCESS CONTROL

There are several role-based access control models in the literature, but the best known is undoubtedly the RBAC96 family of models due to Sandhu *et al* [16], which consists of four role-based access control models of varying degrees of expressive power. All but the simplest model of this family assume the existence of a role hierarchy. It is generally assumed that the role hierarchy supports two different types of inheritance: permissions are inherited upwards and the set of roles available to a user is aggregated downwards. For example, if role r is less senior than r', then any permission assigned to r is implicitly assigned to r', and any user assigned to r' can activate r. To the author's knowledge, most role-based access control models and systems adopt a similar approach to RBAC96 with respect to the role hierarchy and inheritance.[1]

More formally, the role hierarchy is either defined as (the graph of) a binary relation $RH \subseteq R \times R$ or simply as the Hasse diagram of the partially ordered set of roles R.[2] Users and permissions are associated with roles by the binary relations $UA \subseteq U \times R$ and $PA \subseteq P \times R$, where U is the set of users and P is the set of permissions.

A user u is *implicitly* assigned to a role r' if there exists $(u, r) \in UA$ such that $r' < r$ [16]. Hereafter we will adopt the following notation: given a partial order X, $y \in Y$ and $Y \subseteq X$, let $\downarrow y = \{x \in X : x \leqslant y\}$ and $\downarrow Y = \{x \in X : \exists y \in Y, x \leqslant y\}$; then we write $R(u) \subseteq R$ for the set of roles explicitly assigned to user u by the user-role assignment relation and $\downarrow R(u)$ for the set of roles assigned (implicitly and explicitly) to u. Similarly, $R(p)$ denotes the set of roles explicitly assigned to a permission p and $\uparrow R(p)$ denotes the set of roles assigned to p. Figure 1 shows a role hierarchy [15] and illustrates the use of the notation. The roles in the set $\uparrow \{\texttt{PE1}, \texttt{QE1}\}$ are contained within the closed curve in the upper part of the diagram; the roles in the set $\downarrow \texttt{ENG2}$ are contained within the other curve.

It is worth noting that a common assumption in RBAC models is that permissions can be aggregated to form more complex permissions. In other words, we can define a partial ordering on permissions, where $p_1 < p_2$ if p_2 includes all the access rights available to p_1. (Dependencies of this sort are found in the access rights supported by certain operating systems: the Multics access right `w` permits both read and write access to an object, whereas the `r` access right only permits read access.) It is rarely noted that this can lead

[1]One exception is OASIS (Open Architecture for Secure Interworking Services) developed at Cambridge University in the United Kingdom.

[2]The Hasse diagram of a partial order X is the (directed) graph of the transitive, reflexive reduction of the order relation on X [3].

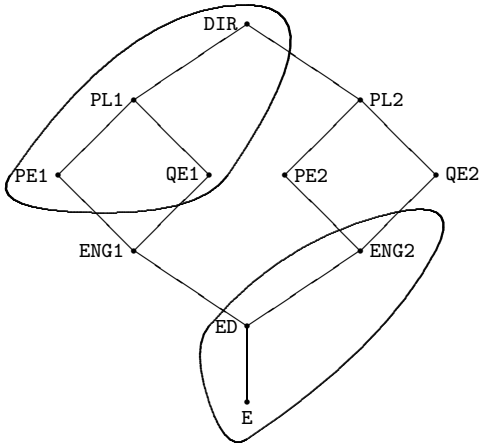


Figure 1: A role hierarchy showing $\uparrow \{\texttt{PE1}, \texttt{QE1}\}$ and $\downarrow \texttt{ENG2}$

to redundancies in the assignment of permissions to roles. In particular, if $p_1 < p_2$ then the set of roles to which p_1 is available should not be a subset of the set of roles to which p_2 is available. Formally, if $p_1 < p_2$ then $\uparrow R(p_1) \not\subseteq \uparrow R(p_2)$; otherwise, the permission-role assignments of p_1 are redundant. We will return to these issues in Section 3.2.

3. A NEW MODEL FOR INHERITANCE IN RBAC

In the next two sections we define the sets and relations used by our model. Our treatment of the role hierarchy and the user-role assignment relation is entirely conventional. Our model diverges from existing approaches when we consider permissions and permission inheritance in Section 3.2.

3.1 Basic features

We assume the existence of a partially ordered set of roles $\langle R, \leqslant \rangle$. We denote the set of edges in the Hasse diagram of R by the binary relation $RH \subseteq R \times R$.

We also assume the existence of a set of users U and define the user-role assignment relation $UA \subseteq U \times R$. If $(u, r) \in UA$ then we say that the user u is *explicitly* assigned to the role r, and that u is assigned to all roles in $\downarrow r$. As before, we denote the set of roles explicitly assigned to u by $R(u)$.

A user's interaction with the system is modelled by a *session*, where a user u activates a subset $S(u)$ of the roles to which he is assigned. That is, $S(u) \subseteq \downarrow R(u)$. A user u may create more than one session, $S_1(u), \ldots, S_k(u)$, where $S_i(u) \subseteq \downarrow R(u)$, $1 \leqslant i \leqslant k$.

3.2 Permissions

We assume the existence of a set of permissions P and define the permission-role assignment relation $PA \subseteq P \times R$. If $(p, r) \in PA$ then we say that the permission p is *explicitly* assigned to the role r. We denote the set of roles explicitly assigned to p by $R(p)$.

We will assume that a permission has the form $(o, \{m_1, \ldots, m_k\})$, where o is an object and m_i, $i = 1, \ldots, k$, is an access mode. We will write $p \leqslant p'$ if $p = (o, M)$ and $p' = (o, M')$ with $M \subseteq M'$.[3] We write $p < p'$ if $p \leqslant p'$ and $p \neq p'$.

[3]We do not wish to elaborate further on the nature of per-

The most important innovation in our model is that each permission is "oriented" (with respect to inheritance) and can be either "up", "down" or "neutral". That is, P is the disjoint union of the sets P^+, P^- and P^0, where P^+ is the set of up permissions, P^- is the set of down permissions and P^0 is the set of neutral permissions.[4]

Each permission $p \in P$ is available to some subset of roles in the hierarchy called the *effective* roles of p and denoted $R_E(p)$. That is, we define the function $R_E : P \to \mathcal{P}(R)$, where

$$R_E(p) = \begin{cases} \uparrow R(p) & \text{if } p \in P^+, \\ \downarrow R(p) & \text{if } p \in P^-, \\ R(p) & \text{if } p \in P^0. \end{cases}$$

The set of permissions *implicitly* assigned to a role r is defined to be $\{p \in P : r \in R_E(p)\}$ and the set of roles to which p is *implicitly* assigned is simply $R_E(p)$.

Note that our model can be interpreted as a standard RBAC model. In this case, all permissions are "up" permissions.

Given a session $S(u) \subseteq \downarrow R(u)$, a request by u to exercise permission p is only granted if u has activated one of p's effective roles; that is, $S(u) \cap R_E(p) \neq \emptyset$.

Informally, we require that the assignment of permissions to roles must reflect the "power" of the permissions. Formally, we require that the following two constraints be satisfied.

CONSTRAINT 1. *If $p < p'$ then either p and p' have the same orientation or $p' \in P^0$.*

CONSTRAINT 2. *If $p < p'$ then $R_E(p) \not\subseteq R_E(p')$.*

Constraint 1 is essentially a consistency constraint, requiring that the direction of inheritance of a permission is consistent with weaker permissions. Constraint 2 is essentially a redundancy check, requiring that a permission is not implicitly assigned to more roles than any weaker permission.

Note that if an alternative model for permissions is used and a different ordering is defined on those permissions, it is still possible and indeed pertinent to insist that Constraint 2 be satisfied. In particular, in the case of standard RBAC models, Constraint 2 means that if $p < p'$ then $\uparrow R(p) \not\subseteq \uparrow R(p')$. (Clearly, Constraint 1 is irrelevant if permission inheritance is uni-directional.) For example, if $r < r'$, $p < p'$ and $(p, r') \in PA$, then the assignment (p', r) introduces redundancy. (How such a violation of Constraint 2 is handled is an implementation matter.)

3.3 Administration

It is self-evident that a role-based access control system is not static: for example, assignments to roles may need to be added or removed and the structure of the role hierarchy may need to be updated in order to reflect organizational changes. It seems reasonable to assume that changes to the *PA*, *UA* and *RH* relations will be performed by roles and hence this area of research is often referred to as *role-based administration*. In short, role-based administration uses roles to control the propagation of permissions.

There have been several attempts to define a role-based administrative model, the most well known being the ARBAC97 model due to Sandhu *et al* [15]. However, we will use the RHA_4 model as the basis of our administrative model [2]. There are several reasons for preferring RHA_4 to ARBAC97 [2], but the primary reason for our choice is that *administrative scope*, the central concept of RHA_4, can be incorporated into our framework more easily than the associated concepts in ARBAC97.

The RHA_4 model defines the `admin-authority` relation, where `admin-authority` $\subseteq R \times R$. If $(a, r) \in$ `admin-authority`, then we say that a is an *administrative role* and r is *controlled* by a; $C(a)$ denotes the set of roles controlled by a. The graph of the `admin-authority` relation can be superimposed onto the role hierarchy to create an extended hierarchy (as shown by the dotted lines in Figure 2).

The *administrative scope* of a role a is a subset of the role hierarchy that can be administered by a. A role r is in the administrative scope of a if any directed path from r and any directed path from any point on that path all pass through a role controlled by a. Informally, this means that any changes made to r do not cause unforseen side effects elsewhere in the hierarchy by virtue of inheritance. Formally, the administrative scope of a, denoted $\sigma(a)$, is defined to be

$$\sigma(a) = \{r \in R : r \in \downarrow C(a), \uparrow r \setminus \uparrow C(a) \subseteq \downarrow C(a)\}.$$

The RHA_4 model can be used to control changes to the *UA* relation and the role hierarchy. In particular,

- the user-role assignment (u, r) can be added to *UA* (by administrative role a) provided $r \in \sigma(a)$;
- the user-role assignment (u, r) can be deleted from *UA* provided $r \in \sigma(a)$;
- an edge (r, r') can be added to the role hierarchy provided $r, r' \in \sigma(a)$;
- an edge (r, r') can be deleted from the role hierarchy provided $r, r' \in \sigma(a)$;
- a role r can be added to R with parents $P \subseteq R$ and children C provided $C, P \subseteq \sigma(a)$;
- a role r can be deleted from R provided $r \in \sigma(a)$.

An illustrative example of administrative scope, based on a hierarchy first used by Sandhu [15], is shown in Figure 2. A dotted edge (r, r') indicates that r controls r'. In particular, `PSO1` controls `PL1` and hence $\sigma(\texttt{PSO1}) = \{\texttt{ENG1}, \texttt{PE1}, \texttt{QE1}, \texttt{PL1}\}$.

In the RHA_4 model the legitimacy of updates to R, *UA*, *RH* and *PA* relations are all determined by the administrative scope of the role making the update. We can adopt most of these aspects of RHA_4 without modification, but we need to review the administration of the *PA* relation because permission inheritance is not uni-directional in our model.

missions: several different possibilities exist and there is no consensus in the research community on which is best. Indeed, some authors prefer to treat permissions as "uninterpreted symbols" [5]. The approach we have chosen enables us to form complex permissions from simpler ones, with the limitation that a permission cannot be defined on more than one object. This approach is well suited to the development of our model as an alternative to the Bell-LaPadula model.

[4]In practice the orientation of a permission (o, M) could be determined by the access modes in M.

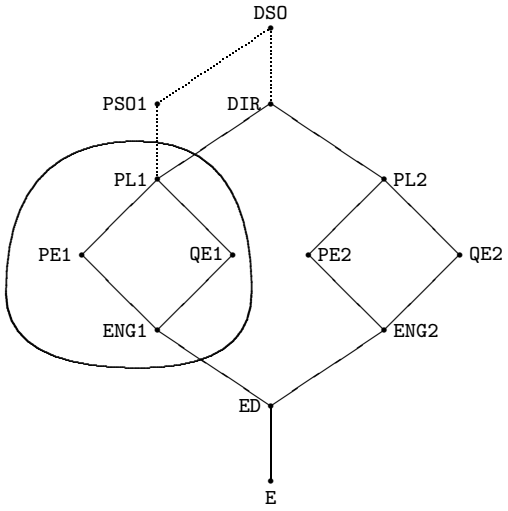


Figure 2: A role hierarchy showing the administrative scope of PSO1

3.3.1 Controlling permission assignment

Every permission p is associated with an effective set of roles $R_E(p)$. Hence we make the following modifications to the RHA_4 model:

- if $p \in P^+$ then (p, r) can be added to or deleted from PA (by administrative role a) provided $r \in \sigma(a)$;
- if $p \in P^0$ then (p, r) can be added to or deleted from PA provided $r \in \sigma(a)$;
- if $p \in P^-$ then (p, r) can be added to or deleted from PA provided $\downarrow r \subseteq \sigma(a)$.

There is a certain asymmetry in the conditions that need to be satisfied for permission-role assignment (or revocation) to take place. In particular, if $p \in P^-$ we require that $\downarrow r = R_E(p) \subseteq \sigma(a)$ (which is a stronger condition than that required when $p \notin P^-$). The reason for this asymmetry is that the concept of administrative scope was introduced specifically to handle inheritance upwards through the role hierarchy.

We now consider a simple example based on the hierarchy in Figure 2. Let $(p, \texttt{PE1}) \in PA$. Then PSO1 can delete this assignment if $p \in P^0$ since $R_E(p) = \{\texttt{PE1}\} \subseteq \sigma(\texttt{PSO1})$. PSO1 can also delete this assignment if $p \in P^+$ since p is inherited in the same way as a permission in a standard RBAC model and the RHA_4 model would permit such a deletion from PA. However, if $p \in P^-$, the deletion of this assignment is not permitted (since $\texttt{ED} \in R_E(p)$ but $\texttt{ED} \notin \sigma(\texttt{PSO1})$, for example). (Note that $\sigma(\texttt{DSO}) = R$ and hence DSO can delete $(p, \texttt{PE1})$ for any p.)

4. ROLE-BASED AND MANDATORY ACCESS CONTROL

The Bell-LaPadula model [1] is probably the most widely known security model and incorporates a mandatory information flow policy [4]. The key features of the Bell-LaPadula model are the *security lattice*, the *simple security property* and the **-property*. Every subject (user) and object is associated with a (security) label, which is an element of the security lattice. The simple security property and *-property require that certain inequalities comparing the security labels of subjects and objects are satisfied.

There have been several attempts to simulate the Bell-LaPadula model using role-based models [9, 13, 14]. The motivation for such research has generally been to demonstrate the versatility and policy neutrality of role-based models; the basic approach has been to make use of inheritance in the role hierarchy.

We now briefly consider three connections between the Bell-LaPadula model and role-based models. The first of these connections is well known in the literature, but, to our knowledge, the others have not been noted previously.

4.1 Permissions

The inheritance of permissions is one reason why it seems attractive to simulate mandatory access control using role-based techniques. Recall that in the standard RBAC models, if $p \in P$ and $(p, r) \in PA$, then p is assigned to all roles in $\uparrow r$. Hence, any user assigned to a role at least as senior as r can exercise permission p. This is analogous to the ability of a subject to read any object whose security label is dominated by that of the subject.[5]

In its simplest form, the *-property requires that the security label of a subject is no greater than that of any object to which the subject tries to write. In other words, if a subject can write to an object, any subject with a lower security label can also write to that object. Since the inheritance of permissions in the role hierarchy is always upwards, role-based access control models do not directly support a simulation of the *-property, which requires that certain permissions be inherited downwards.

Role-based approaches have circumvented this problem by introducing two partial orderings, $\leqslant_r$ and $\leqslant_w$, on the set of roles, where $r \leqslant_r r'$ if and only if $r' \leqslant_w r$. This gives rise to two hierarchies RH_r and RH_w to which read and write permissions are respectively assigned [13].[6]

However, this begs the question"How should read/write permissions be handled?". In Section 4.4 we show how our approach to permission inheritance can be used to simulate the behaviour of the Bell-LaPadula model using a single hierarchy. Furthermore, it is possible to assign read/write permissions to roles in the hierarchy.

4.2 Security labels and role hierarchies

In this section we examine why the set of roles assigned to a user can be interpreted as a security label for that user. We also discuss why the set of roles assigned to a permission does not have the same interpretation and briefly consider how we might define suitable security labels for permissions and objects in a role-based model.

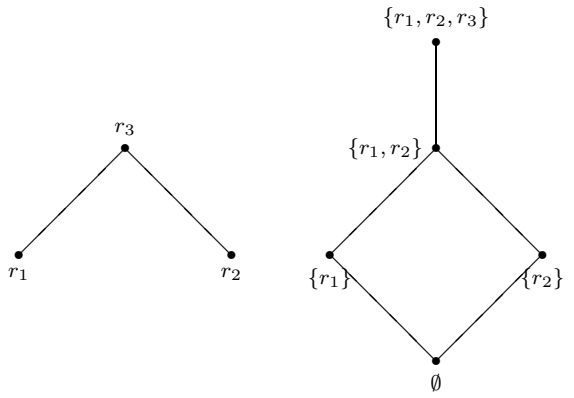
Let $\langle R, \leqslant \rangle$ be a partially ordered set of roles and let $R(u)$

[5]The analogy is not exact because the Bell-LaPadula model permits access only if the simple security property is satisfied *and* the relevant entry exists in the protection matrix. In this sense, any role-based simulation of Bell-LaPadula assumes that the protection matrix contains every possible access right in each cell, which is hardly a realistic assumption. Nevertheless, the simple security property is accurately captured in role-based models.

[6]Formally, $\langle R, \leqslant_w \rangle$ is the *dual* of $\langle R, \leqslant_r \rangle$ [3]. Informally, the Hasse diagram of $\langle R, \leqslant_w \rangle$ is obtained by inverting the Hasse diagram of $\langle R, \leqslant_r \rangle$.

denote the set of roles explicitly assigned to the user u. Then $R(u)$ can be regarded as a security level in a suitable security lattice. In particular, let $\mathcal{I}(R) = \{\downarrow S : S \subseteq R\}$. $\langle \mathcal{I}(R), \subseteq \rangle$ is the lattice of *order ideals* in R [3]. Then for any user u we associate the set $R(u)$ with the "security label" $\downarrow R(u)$ in $\mathcal{I}(R)$.

For example, Figure 3 depicts a role hierarchy in which $R = \{r_1, r_2, r_3\}$, $r_1 < r_3$ and $r_2 < r_3$. The antichains in $R(u)$ are $\emptyset$, $\{r_1\}$, $\{r_2\}$, $\{r_3\}$ and $\{r_1, r_2\}$, which correspond to the sets $\emptyset$, $\{r_1\}$, $\{r_2\}$, $\{r_1, r_2, r_3\}$ and $\{r_1, r_2\}$, respectively.[7]



(a) $\langle R, \leqslant \rangle$ (b) $\langle \mathcal{I}(R), \subseteq \rangle$

Figure 3: A role hierarchy and the associated lattice of order ideals

It is generally assumed that a user u activates a session $S(u) \subseteq \downarrow R(u)$ using some subset of the roles assigned to u. The Bell-LaPadula model is usually extended to include the concept of a *current security function*, which enables a privileged user to downgrade his security level, thereby allowing him to write to less privileged objects. Hence we can regard the label $\downarrow S(u)$ as the *current security label* of u. For example, a user assigned to r_3 can create a session using role r_1, thereby creating a current security label of $\{r_1\}$.

However, the concept of a security level for a permission is more problematic. RBAC models generally assume the existence of a permission-role assignment relation and that a user u can use permission p if there exist roles r and r' such that $(u, r) \in UA$, $(p, r') \in PA$ and $r' \leqslant r$; in other words, the criterion for permission usage is "existential" with respect to permission-role assignment. However, in the Bell-LaPadula model the criterion is "universal" – the security label of a subject must dominate every part of the security label of an object. In other words, if we defined the security level of a permission to be $R(p)$, we would require that $R(p) \leqslant R(u)$, where this order inequality is interpreted as $\downarrow R(p) \subseteq \downarrow R(u)$.

In general, therefore, the requirements of permission usage are incompatible with the requirements of object access in the Bell-LaPadula model. The two sets of requirements converge when $R(p) = \{r\}$ for some $r \in R$; that is, there exists a permission-role assignment *function*. In this case, the security level of the permission is $\downarrow r$. However, this is likely to be too restrictive a requirement in general RBAC models.

The concept of a security level for an object in RBAC is also problematic. We could insist that all permissions for a particular object are assigned to the same role r. (In this case, the security level of the object is $\downarrow r$.) However, this definition is rather limiting; we discuss an alternative in Section 4.4.2.

4.3 The *-property and separation of duty

Let $\lambda(o)$ denote the security level of an object o and let o_r and o_w be two objects in the Bell-LaPadula model. Then the *-property states that a subject s that currently has read access to o_r will only be permitted write access to o_w if $\lambda(o_w) \geqslant \lambda(o_r)$. That is, a system that implements the Bell-LaPadula model necessarily implements what might be called dynamic permission-based separation of duty. In particular, whenever $\lambda(o_w) < \lambda(o_r)$, the *-property will prevent any user obtaining read access to o_r and write access to o_w contemporaneously. The purpose of this feature of the *-property is to prevent information flow from more secure objects to less secure objects, thereby providing *inter alia* protection against Trojan horses.

4.4 Simulating the Bell-LaPadula model

In this section we demonstrate how, with the addition of a few constraints, our model can be used to simulate the Bell-LaPadula model. The simplicity of this simulation compares favourably with previous attempts, which typically require changes or additions to the underlying RBAC model (such as the inclusion of a second role hierarchy). Our approach also supports the assignment of "mixed" permissions, which include both read and write access to permissions. A detailed comparison of our approach with earlier research is beyond the scope of this paper.

4.4.1 Permissions

The Bell-LaPadula model makes a distinction between read access modes and write access modes because of the requirements of the information flow policy it enforces. In particular, the `w` access right in the Multics implementation of the Bell-LaPadula model provides simultaneous read/write access to an object.

For each object o we assume there exists a set of access modes $\mathcal{M}$ and that $\mathcal{M} = \mathcal{M}_r \cup \mathcal{M}_w$, where $\mathcal{M}_r$ and $\mathcal{M}_w$ are read and write modes respectively. This set of access modes will typically vary from object to object. It is not necessarily true that $\mathcal{M}_r \cap \mathcal{M}_w = \emptyset$. That is, some access modes can be both read and write modes. For example, a complex mode might invoke *get* and *set* methods on the object.

DEFINITION 1. *A permission of the form $(o, \{m\})$, where $m \in \mathcal{M}$ is called an* atomic *permission. A permission (o, M_r), where $M_r \subseteq \mathcal{M}_r$, is called a* read *permission; a permission (o, M_w), where $M_w \subseteq \mathcal{M}_w$, is called a* write *permission; a permission is* simple *if it is either a read or write permission; and a permission is* compound *if it is not simple.*

A permission p is minimal *if there exists $r \in R$ such that $(p, r) \in PA$ and for all $p' \leqslant p$, if $(p', r') \in PA$ then $r' = r$.*

[7]The RBAC96 model does not require that the set of roles explicitly assigned to u be an antichain; nevertheless, the mapping to a security label is still valid. For example, if $R(u) = \{r_1, r_3\}$, then $\downarrow R(u) = \downarrow r_3 = \{r_1, r_2, r_3\}$.

A permission p is maximal *if there exists $r \in R$ such that $(p, r) \in PA$ and for all $p' \geqslant p$, if $(p', r') \in PA$ then $r' = r$.*

A consequence of the definition is that any minimal permission (or maximal permission) is assigned to a unique role. Note that there may be more than one minimal permission associated with an object. For example, we could assign atomic permissions $p = (o, \{m\})$ and $p' = (o, \{m'\})$ $(m \neq m')$ to `PE1` and `QE1` respectively; then, by definition, p and p' are minimal permissions.

4.4.2 Constraints

In order to simulate the Bell-LaPadula model our role-based model must be constrained in the following ways:

CONSTRAINT 3. *PA is a (partial) function.*

In other words, every permission is assigned to a unique role. The justification for this constraint is provided in Section 4.2. The security level of p is defined to be $R(p)$, which equals $\{r\}$ for some $r \in R$.

CONSTRAINT 4. *For each object o, there exists a unique minimal read permission p_r and a unique maximal write permission p_w.*

Hence there exist roles $r_{\min}$ and $r_{\max}$ such that $(p_r, r_{\min}) \in PA$ and $(p_w, r_{\max}) \in PA$. The *security level* of the object is defined to be the range

$$[r_{\min}, r_{\max}] = \begin{cases} \{r \in R : r_{\min} \leqslant r \leqslant r_{\max}\} & \text{if } r_{\min} \leqslant r_{\max} \\ \emptyset & \text{otherwise.} \end{cases}$$

CONSTRAINT 5. *If $p' < p$ then $R_E(p) \subset R_E(p')$.*

This constraint is stronger than Constraint 2, which was formulated to prevent redundant permission-role assignments. In the case of mandatory access control, we require a stronger constraint, which requires that a permission is available only to a strict subset of the effective role sets of weaker permissions.

CONSTRAINT 6. *If p is a write permission, then $p \in P^-$.*

If p is a write permission then for all $p' < p$, p' is a write permission and Constraint 5 requires that $R_E(p) \subset R_E(p')$. Hence for all write permissions p, p' such that $p < p'$, $R(p) > R(p')$.

CONSTRAINT 7. *If p is a read permission, then $p \in P^+$.*

If p is a read permission then for all $p' < p$, p' is a read permission and Constraint 5 requires that $R_E(p) \subset R_E(p')$. Hence for all read permissions p, p' such that $p < p'$, $R(p) < R(p')$.

CONSTRAINT 8. *If p is a compound permission, then $p \in P^0$.*

PROPOSITION 1. *A compound permission $p = (o, M)$ can only be assigned to a role within the security level of o.*

PROOF. Let $p = (o, M_r \cup M_w)$, where $\emptyset \neq M_r \subseteq \mathcal{M}_r$ and $\emptyset \neq M_w \subseteq \mathcal{M}_w$. Then by Constraint 5, $R_E(p) \subseteq R_E(p_r)$ and $R_E(p) \subseteq R_E(p_w)$, where $p_r = (o, M_r)$ and $p_w = (o, M_w)$; that is,

$$\begin{aligned} R_E(p) &\subseteq R_E(p_r) \cap R_E(p_w) \\ &= \uparrow R(p_r) \cap \downarrow R(p_w) \\ &\subseteq \uparrow r_{\min} \cap \downarrow r_{\max} \\ &= [r_{\min}, r_{\max}]. \end{aligned}$$

Now $p \in P^0$ and hence, by Constraint 8, $R_E(p) = R(p)$. Therefore, $R(p) \in [r_{\min}, r_{\max}]$. □

If the security level of an object is $\emptyset$, no role can both read and write that object. This may be useful for an object such as an audit file f: low level roles (must) write to f, while high level roles can read f but must not be able to change it.

CONSTRAINT 9. *No user can run more than one session at a time. The set of roles in a session must form an antichain.*

4.4.3 Example

A formal comparison of our role-based model and the Bell-LaPadula model is beyond the scope of this paper. However, we will briefly consider an illustrative example using a simple set of access modes.

Let $\mathcal{M} = \{m_r, m_w\}$, where m_r is a read access mode and m_w is a write access mode. Let $p_r = (o, \{m_r\})$ be a read permission assigned to r_1 and let $p_w = (o, \{m_w\})$ be a write permission assigned to r_2. (Hence, p_r and p_w are minimal permissions provided neither permission is assigned to any other role.) Then $R_E(p_r) = \uparrow r_1$ and $R_E(p_w) = \downarrow r_2$.

The example is summarised in Figure 4. Figure 4a shows the partial ordering on permissions; Figure 4b shows the permission-role assignment relation; the three remaining sub-figures show different role hierarchies. We write $\lambda(o)$ to denote the security level of o.

A user u can only use permission p_r if u can activate a session containing a role r', where $r' \geqslant r$. In other words, u can use permission p_r only if $R(u) \geqslant R(p)$. Similarly, u can use permission p_w only if u can activate a role r', where $r \leqslant r'$.

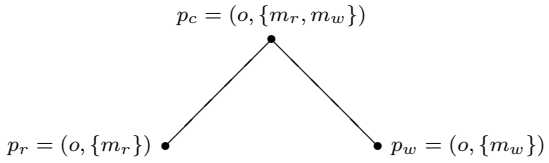
If $p_c = (o, \{m_r, m_w\})$ is a compound permission, then p_c can only be assigned to a role in the range $[r_1, r_2]$ (if it exists). In Figure 4c, p_c can be assigned to either r_1 or r_2. However, in Figure 4d, p_c can not be assigned to any role. A user u can use permissions p_r and p_w only if u can activate role r_1 and r_2 in the same session. (Separation of duty constraints can be specified to prevent any user from activating r_1 and r_2 if necessary.)

Note that if $r_1 = r_2$ and hence $(o, \{m_r\})$ and $(o, \{m_w\})$ are assigned to the same role, then the security level of o is $\{r_1\}$, $R_E(p) = \{r_1\}$ and hence u must activate r_1 to use permission p. This is analogous to the situation in the Bell-LaPadula model where a user must have the same security level as an object in order to read and write to it.

4.5 Administration

One problem with the Bell-LaPadula model has been the difficulty of maintaining a secure system when arbitrary changes can be made to security levels.

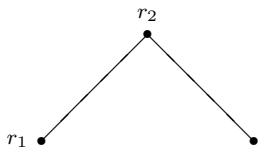
In particular, McLean showed that a secure system as defined by Bell-LaPadula does not necessarily imply that the system exhibits secure behaviour in any practical sense. He



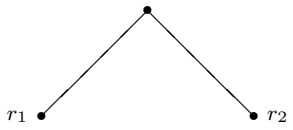
(a) Permission hierarchy

PA	
p_r	r_1
p_w	r_2

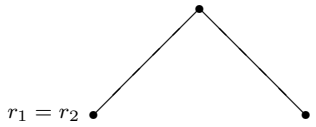
(b) Permission assignments



(c) $\lambda(o) = [r_1, r_2]$



(d) $\lambda(o) = \emptyset$



(e) $\lambda(o) = \{r_1\}$

Figure 4: Security levels using RBAC

suggested a solution in which changes to the security function were made in a controlled way [10]. Specifically: let S be the set of subjects and O be the set of objects, with $S \subseteq O$; we define a function $\psi : S \cup O \to \mathcal{P}(S)$, where $\psi(o)$ determines the set of subjects that can change the security level of o; a state transition is assumed to be initiated by a subject s and is said to be *transition secure* only if any changes to the security function are permitted by the function ψ. In other words, the subject s must be authorized (by ψ) to change the security level of o. (If $\psi(o) = \emptyset$ for all o, then the security function cannot be changed. Such a system is said to have the property of *tranquillity*.)

In the context of role-based access control, we assume that each change to the configuration of the system is made by a role (which is analogous to a subject). Using our administrative model we have a natural implementation of ψ using the concept of administrative scope. Specifically, a role r can change user-role and permission-role assignments (the security levels of subjects and objects) for all roles $r' \in \sigma(r)$. In other words, σ is a natural choice for ψ. The simplest implementation is to let the set of roles controlled by a role r be $\{r\}$. In this case, $\sigma(r) = \{s \in \downarrow r : \uparrow s \setminus \uparrow r \subseteq \downarrow r\}$. Using the hierarchy in Figure 2, for example, $\sigma(\texttt{PL1}) = \{\texttt{ENG1}, \texttt{QE1}, \texttt{PE1}, \texttt{PL1}\}$ and $\sigma(\texttt{DIR}) = R$.

5. CONCLUSIONS AND FUTURE WORK

We have defined a model with a different approach to permission inheritance. The obvious questions to ask are: "Does this approach address any of the perceived shortcomings of the standard RBAC approach to inheritance?"; and "Does this approach provide any tangible advantages over the standard approach?".

In answer to the first question, we note that it is possible to implement separation of duty constraints on two roles that have a common senior role and for a user to activate the senior role; it requires that all sensitive permissions assigned to the two junior roles are neutral permissions. This is not possible in standard RBAC. However, our approach does not prevent a senior role having access to all possible permissions because we have not placed any restrictions on the roles that a user can activate. (Recall that a user u may activate any role in $\downarrow R(u)$.) Therefore, we could augment our model with the concept of a *user-role-assignment floor*, which could be modelled as an antichain $F(u)$ in R. Then a user u could only activate roles in the set $\downarrow R(u) \setminus \downarrow F(u)$. Using the hierarchy in Figure 2, for example, we could imagine assigning a floor of $\{\texttt{PE1}, \texttt{QE1}, \texttt{PE2}, \texttt{QE2}\}$ to any user who is assigned to `DIR`. Such a user would only be able to activate the (senior) roles `PL1`, `PL2` and `DIR`. This is an area for further research.

In answer to the second question, we note that our approach provides considerably more flexibility than standard role-based models. (It would appear that our approach does not introduce any significant additional overheads. In existing role-based models it is necessary to traverse the role hierarchy in both an upward and downward direction: in the former case to determine the roles implicitly assigned to a permission; and in the latter to determine the roles implicitly assigned to a user.)

We also note that our approach provides a more direct implementation of the Bell-LaPadula model using role-based techniques. Our approach also provides a rigorous definition of the security level of a permission and of an object. Moreover, we do not require an additional hierarchy and are able to assign complex permissions to roles.

A useful side effect of our study of permission inheritance has been the definition of constraints that all role-based models should enforce. These constraints are necessary whenever there exists an (implicit or explicit) ordering on the set of permissions in order to provide consistency and prevent redundancy in the permission-role assignment relation. Moreover, such constraints provide a useful insight into the form of constraints that would be required when simulating the Bell-LaPadula model.

The natural extension of the ideas in this paper is to develop a formal statement of the security properties of our role-based model when constrained in the ways described in Section 4.4.2. In short, can we develop a result analogous to the "Basic Security Theorem" for the Bell-LaPadula

model [1]? It is this question that will occupy our research in the short term.

Acknowledgements. I am grateful to Kenny Paterson and George Loizou, whose helpful comments contributed to the readability of this paper.

6. REFERENCES

[1] Bell, D., and LaPadula, L. Secure computer systems: Unified exposition and Multics interpretation. Tech. Rep. MTR-2997, Mitre Corporation, Bedford, Massachusetts, 1976.

[2] Crampton, J., and Loizou, G. Administrative scope: A foundation for role-based administrative models. *ACM Transactions on Information and System Security 6*, 2 (2003). 201–231.

[3] Davey, B., and Priestley, H. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, United Kingdom, 1990.

[4] Denning, D. A lattice model of secure information flow. *Communications of the ACM 19*, 5 (1976), 236–243.

[5] Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D., and Chandramouli, R. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security 4*, 3 (2001), 224–274.

[6] Gavrila, S., and Barkley, J. Formal specification for role based access control user/role and role/role relationship management. In *Proceedings of Third ACM Workshop on Role-Based Access Control* (Fairfax, Virginia, 1998), pp. 81–90.

[7] Goh, C., and Baldwin, A. Towards a more complete model of role. In *Proceedings of Third ACM Workshop on Role-Based Access Control* (Fairfax, Virginia, 1998), pp. 55–61.

[8] Kuhn, D. Mutual exclusion of roles as a means of implementing separation of duty in role-based access control systems. In *Proceedings of Second ACM Workshop on Role-Based Access Control* (Fairfax, Virginia, 1997), pp. 23–30.

[9] Kuhn, D. Role based access control on MLS systems without kernel changes. In *Proceedings of Third ACM Workshop on Role-Based Access Control* (Fairfax, Virginia, 1998), pp. 25–35.

[10] McLean, J. Security models. In *Encyclopedia of Software Engineering*, J. Marciniak, Ed. John Wiley & Sons, 1994.

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

[12] Nyanchama, M., and Osborn, S. The role graph model and conflict of interest. *ACM Transactions on Information and System Security 2*, 1 (1999), 3–33.

[13] Osborn, S., Sandhu, R., and Munawer, Q. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security 3*, 2 (2000), 85–106.

[14] Sandhu, R. Role hierarchies and constraints for lattice-based access controls. In *Proceedings of Fourth European Symposium on Research in Computer Security* (Rome, 1996), pp. 65–79.

[15] Sandhu, R., Bhamidipati, V., and Munawer, Q. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and System Security 1*, 2 (1999), 105–135.

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