A Model of OASIS Role-Based Access Control and its Support for Active Security

Walt Yao
University of Cambridge
Computer Laboratory
Pembroke Street
Cambridge
United Kingdom
walt.yao@cl.cam.ac.uk

Ken Moody
University of Cambridge
Computer Laboratory
Pembroke Street
Cambridge
United Kingdom
ken.moody@cl.cam.ac.uk

Jean Bacon
University of Cambridge
Computer Laboratory
Pembroke Street
Cambridge
United Kingdom
jean.bacon@cl.cam.ac.uk

ABSTRACT

OASIS is a role-based access control architecture for achieving secure interoperation of services in an open, distributed environment. Services define roles and implement formally specified policy for role activation and service use; users must present the required credentials, in the specified context, in order to activate a role or invoke a service. Roles are activated for the duration of a session only. In addition, a role is deactivated immediately if any of the conditions of the membership rule associated with its activation becomes false.

OASIS does not use role delegation but instead defines the notion of appointment, whereby a user in some role may issue an appointment certificate to some other user. The role activation conditions of services may include appointment certificates, prerequisite roles and environmental constraints.

We motivate our approach and formalise OASIS. First, a basic model is presented followed by an extended model which includes parameterisation.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; D.4.7 [**Operating Systems**]: Organization and Design—*Distributed systems*

General Terms

Design, Security, Theory

Keywords

Role based access control, RBAC, OASIS, service level agreements, certificates, policy

1. INTRODUCTION

The growing interest in role-based access control (RBAC) as an effective means of replacing traditional discretionary and mandatory access control has led to the development of several models over the past few years [20, 8, 16]. Besides the basic structure of subject, role and privilege, they all include a notion known as *role hierarchy*. In its original interpretation, a senior role in a hierarchy is treated as an instance of all its junior roles. This means a senior role is granted all privileges given to junior ones.

Although role hierarchy has become part of the accepted model, we question whether it is appropriate in real-world applications. In general a senior role will not need all the privileges of its junior roles to carry out its work, and this leads to violation of the principle of least privilege [18]. One of the main reasons for using RBAC is that it provides a natural way to model constraints such as separation of duties. Role hierarchies complicate the specification and enforcement of these constraints. For example, one kind of separation of duties constraint is that a subject cannot have a pair of conflicting privileges at any time. Privileges are assigned to roles, and the interpretation of this constraint is that a subject cannot act in a pair of conflicting roles simultaneously. Suppose that this pair of roles is in a hierarchy, each being subsidiary to the same senior role. It would then be impossible to exercise the senior role without breaking either the constraint or the hierarchy.

There are various proposals to work around these problems. Sandhu [19] proposes separating the purposes of a hierarchy into usage and activation. In an activation hierarchy a user may choose to activate any role that is below her assigned role in the hierarchy, therefore avoiding having two conflicting roles active simultaneously. Ferraiolo et al. [8] choose to override the inheritance relationship whenever there is tension between hierarchy and conflict. Moffett [14] suggests employing an ordering other than the organisational hierarchy to define a role hierarchy, or to use subsidiary roles (also known as *private roles* [20]) outside the organisational hierarchy. Such solutions are awkward since they require an organisation to adapt its security policies in order to avoid the potential problems of role hierarchies. Moffett and Lupu [15] examine some possible uses of role hierarchies and identify three potential interpretations for role hierarchies, namely isa hierarchy, activity hierarchy, and supervision hierarchy. We believe that the use of role hierarchies arises mainly through the influence of object-oriented modelling and we are not convinced of their utility in practice.

Goh questions the concept of role hierarchies from the point of view of subsidiarity in [10]. He argues that useful role hierarchies are uncommon in a real organisation where tasks are assigned to appropriate roles independently of the authority structure. We believe this view is a step forward towards a more practical model for role based access control.

In this paper, we present a role based access control model in which the fundamental role-role relationship, *role activation dependency*, is dynamic. Each role activation is governed by a set of rules, which are specified in logic. Roles are parameterised, which helps to make policy expression more scalable. Our model offers several advantages over existing ones while retaining most of their desirable features. The advantages include: (1) each role is named by a specific issuing service, so that it is easy to define roles and establish policies for each service independently. (2) role activation is controlled by rules which need to change only if the underlying security policies change, so it is possible to deploy policy separately for each administrative domain. (3) the use of parameters makes it easy to tailor the model to specific applications, since state may be read from the environment when each role is activated. In this way active security based on tasks or workflow can be implemented naturally. (4) our logic-based approach supports formal reasoning about policy.

The remainder of this paper is organised as follows. Section 2 introduces our model informally and relates it to the literature. Section 3 discusses *appointment* which replaces delegation in the model. Section 4 provides a formal description of the model. We present a simplified model in detail, and indicate how it can be extended to include parameterisation. Section 5 describes a scenario to demonstrate one application of the model. Section 6 concludes this paper with a summary.

2. THE OASIS ACCESS CONTROL MODEL

OASIS stands for Open Architecture for Secure Interworking Services. It is designed to facilitate access control in distributed systems. OASIS embodies an open, decentralised approach; for example, roles are defined by services and services may interoperate, recognising one another's roles, according to service-level agreements. We do not envisage a single centralised role administrator. Previous work on OASIS has focused mainly on the architectural issues [11, 3]. [12] discussed engineering issues in large-scale OASIS implementations. It is important to note that OASIS is integrated with an event infrastructure [3]; this allows services protected by OASIS to communicate asynchronously, so that one service can be notified immediately of a change of state at another. This paper addresses the formalisation of the OASIS model, but first we motivate the essential concepts on which it depends.

Central to the OASIS model is the idea of credential-based role activation. The credentials that a user possesses, together with side conditions which depend on the state of the environment, will authorise him or her to activate a number of roles. At any given time, a user will activate a subset of these potential roles in order to carry out some specific task, thus embodying the principle of least privilege in an organisation [18]. The ability to activate and deactivate roles is vital to the support for active security [23], where the context is taken into consideration when an access is requested. The concept of role activation in OASIS is similar to the concept of session in [20], except that a user cannot deactivate at will. Activation of any role in OASIS is explicitly controlled by a *role activation rule*, and this rule may require that specified preconditions continue to hold while the role remains active, the *role membership rule*. When a role is activated at a service an event channel is created in association with each membership condition. An event is triggered immediately any such condition becomes false, causing the role to be deactivated. For example, such a trigger may be generated when a timer expires or when a database is updated to remove a user from membership of a group.

A role activation rule specifies the conditions that a user must meet in order to activate a role. The intuition behind this is that roles are usually given to a person provided that he or she has met certain conditions, e.g. being qualified as a physician, being employed by a company, being assigned to a task, being on shift, etc. We model these conditions in three categories, namely: prerequisite roles, appointments, and constraints.

A prerequisite role in the condition for a target role means that a user must have already activated the prerequisite role before he or she can activate the target role. This is a session-based notion. The basis of the concept of prerequisite roles is competency and appropriateness, as pointed out by Sandhu et al. [20].

Appointment occurs when a member of some role grants a credential[1] that will allow some user to activate another role. The context may be an assignment of jobs or tasks. It may also be the passing of an examination, becoming professionally qualified, or becoming employed. An activation rule for a role requiring an appointment will require that the associated credential is presented when activating that role. Note that the appointing role is in no sense delegating; a clerk in a hospital registry will not be medically qualified.

Security policies in real life often involve constraints such as separation of duties. Several types of constraint have been identified and discussed in the literature [20, 13, 21, 17]. In our model, constraints may be associated with role activation rules, see definition 4 in section 4.1; in future work we plan to specify role constraints at the organisational level, for example, "an account clerk cannot simultaneously be a billing clerk". We describe a possible implementation of role constraints in the discussion of negated prerequisite roles following definition 6.

The use of roles allows access control policy to be specified in terms of the privileges of categories of users. This has two advantages: first, there is no need to change policy as staff come and go; second, details of individuals need only be taken into account during role activation. On the other hand, we feel that in many applications it is insufficient to base access control decisions solely on roles and their assigned privileges. This is especially true when context information such as time needs to be considered. Extensions have been proposed to the basic RBAC models in order to support workflow systems [6], team-based systems [22, 24], and, more generally, in the content-based access control model of

[1]For clarity we introduce a new term *appointment certificate* for the credential associated with an appointment. We had earlier referred to *auxiliary credential certificates*, see [3].

Giuri and Iglio [9] and the generalised model of Covington et al.[7]. Most of these proposals are specific to their application domains.

In OASIS, we have extended the role model with parameters, based on first-order logic. Parameters may be included in the rules that cover both role activation and access to an object or service. Parameters may be bound to such items as the time of a role activation, the userid of a file owner, or an attribute of the object that is being accessed. The values that instantiate parameters are therefore context-dependent. Our model is similar to some extent to Giuri and Iglio's model based on role templates [9]. The main difference is that our model uses formal first-order predicate logic as its foundation.

In section 4.1 we present the details of a simplified model using propositional logic. This version omits parameters, but it is identical to the full model in all other essential features. In section 4.2 we outline the extensions necessary to include parameters. We adopt a formal approach using logic because adding parameters to privileges and roles adds a layer of complexity to the model. A logic-based approach helps to reduce errors in security policies by allowing static checks to be performed, for example, for completeness, consistency and reducibility. It also allows formal reasoning about security policies to discover potential errors or conflicts. Furthermore, the use of logic enables our model to be integrated with policies specified in pseudo-natural language. Preliminary, proof-of-concept work in this area can be found in [2].

3. APPOINTMENT

Role delegation is an extension to the conventional delegation of privileges, in which one user grants privileges to another through roles. Recent and significant work in this area has been done by Barka and Sandhu. In [5], a role-based delegation model called RBDM0 is introduced. In RBDM0, if a role is delegated then all the associated privileges are granted. Delegation is limited to a single step. Later work by the same authors [4] identifies cases of role-based delegation that are useful in practice, and in particular extends the model to include cascading delegation.

We introduce the notion of *appointment* to replace delegation. A user in a role acting as an appointer grants another user, the appointee, a credential which may be used to activate one or more roles. Roles activated on the basis of an appointment are usually associated with some tasks or responsibilities, and encapsulate the privileges granted by the appointer.

3.1 Appointment vs. Delegation

The appointment model offers several advantages over the traditional delegation model. First, privilege propagation is controlled and well-defined. In [4], an attribute *totality* is introduced to indicate whether the whole set of privileges assigned to a role are to be delegated, or only some subset. The latter case is known as partial delegation. Partial delegation breaks the formal semantics of role-based access control since a partially delegated role is in fact a new role sharing an overloaded name with its delegating role. In the appointment model, only those roles required to complete a job function may be activated by the appointee, and conditions may be specified to restrict the context of activation. The appointment model thus goes some way towards embodying the principle of least privilege [18]. Appointment in itself confers no privileges. Any privileges derive solely from roles activated on the basis of an appointment, and are limited to the current session. Privileges are independent of the appointer role.

Second, cascading delegation becomes irrelevant since appointees are typically in a different role from their appointer. The level of delegation attribute in [4] therefore becomes redundant.

Third, in the appointment model, an appointer can give access to privileges that he or she does not possess, albeit in a controlled fashion. There is no reason why an appointer should be able to satisfy the conditions for activating a role that is to be granted to the appointee.

In general, delegation can be viewed as a special case of appointment, in which a user in some role may appoint another user to activate that same role. This mechanism could be part of the emergency procedure of a service when a role holder is called away or is taken ill.

3.2 Taxonomy

Barka and Sandhu describe a classification framework for role-based delegation models in [4]. While some of these attributes are useful and indeed essential when designing a delegation model, we present a different taxonomy specific to role-based appointment and dynamic role activation. Our taxonomy involves three types of users, appointer, appointee and revoker. The appointer is someone with the credentials for activating the appointing role, the appointee specifies who is to be allocated an appointment, and the revoker specifies who can revoke an appointment.

Appointer. Appointment implies the granting of privileges. It is therefore essential to restrict which users may grant each type of appointment. For each specific type of appointment some role is identified to be the appointer. So long as a user is acting in the appointer capacity, i.e. the user has activated the appointer role, the user can make appointments of the given type. For most organisational policies, it is sufficient to control who may appoint through an appointer role. If it is necessary to restrict those granting appointments to a specific set of users, this can be enforced by side conditions when the appointer role is activated.

Appointee. An appointment is granted so that the appointee can obtain privileges by activating some role. At the least the activation rules will require that the user is known to the system, for example through holding a credential as an authenticated user. In some circumstances that may be the only requirement. An example is that a doctor on duty in Accident and Emergency (A&E) may appoint any member of the hospital staff who is on duty to order a blood test for a patient, rather than being constrained to appoint only a nurse in the A&E team.

If on the other hand the appointment is to be long-lived, the regulations that govern the issue of appointment certificates can specify checks that should be made on all appointees. An example is when a new doctor is registered as an employee of a hospital. The administrator making an appointment (and/or issuing a smartcard) must check the new doctor's academic and professional qualifications. Checks may include a mixture of clerical and computational procedures. Provided that the checks are satisfied the appointer will apply for an appointment certificate to transfer to the appointee. A revocation credential specific to the appointment is issued to and may be retained by the appointer, see below.

It is possible to restrict the use of appointment certificates independently of their issue. In particular, an appointment certificate may require that the user presenting it is already active in one or more roles, see definition 3. Such a condition is appropriate when an appointment should only be activated by staff who are already on duty. The appointment certificate may also be subject to predicates which can include environmental constraints, see definition 11 in section 4.2.

Revoker. In OASIS an appointment can be revoked in three possible ways: by its appointer only; by anyone in the appointer role; or by rule-based system revocation. The first two cases make use of the revocation credential returned at the time of appointment.

In the first case, an appointment can only be revoked by its appointer. This is common in real-life organisations; for example, the lead doctor in a care team might assign tasks to staff on that team by means of appointment. He then becomes responsible for monitoring performance. Revoking the appointment of any member who performs badly is up to the lead doctor himself. The revocation credential is valid only when presented by the user who made the appointment. This is called *appointer-only revocation*.

As pointed out in [5], dependence on a particular user to revoke may have undesirable consequences; for example, if an appointer is on leave it may be impossible to take immediate action to limit damage. A solution is to allow anyone who can activate the appointer role to make the revocation. This is called *appointer-role revocation*. This is helpful only if more than one user can activate the appointer role. The principle is to limit the spread of damage by increasing the number of people who can stop a misbehaving party.

The third possibility for revocation is *system-managed revocation*. In this case, an appointment is revoked automatically if certain conditions are met. There are many circumstances in which the revocation of an appointment can be better handled by the system than by a human. Continuing the A&E scenario, the lead doctor may appoint a nurse to order a blood test and wish that appointment to be revoked as soon as the order has been successfully made.

Three possible types of system-managed revocation are based on time, tasks and sessions. For time-based revocation, an appointment is associated with an expiry time. The appointment certificate is revoked automatically at the expiry time. This is appropriate if the policy is to review long-lived credentials at regular intervals. In other applications, such as an appointment to complete a specific job, it may be difficult to apply in practice. An alternative approach is to bind an appointment to its assigned responsibilities, expressed as tasks. The system monitors the progress of these tasks and once they have been completed successfully, the appointment is automatically revoked. This approach, which requires substantial support from a task model, is suitable in a workflow environment.

The third type of system-managed revocation is based on *sessions*. This can have two interpretations, the session of the appointer or the session of the appointee. In the former case, an appointment is valid so long as the appointer role is still activated. It will be revoked automatically when the appointer leaves the appointer role. This type of revocation ensures tight monitoring of the appointee. An appointment can also be for the duration of the current session of the appointee. For example, a junior doctor may be appointed to stand in for a consultant who is called away to an emergency. When the junior's shift is over he logs off and the session and appointment end. In practice the membership rules for a role entered by an appointee will often require that some other role remain active, and use of the appointment is therefore limited to the associated session.

When designing practical systems which deploy the OASIS model we have used appointment for long-standing persistent credential allocation. Examples are credentials that depend on academic or professional qualifications, or on holding a particular job in an organisation. Such credentials are used, among others, to activate roles in order to carry out tasks for the duration of a session.

4. FORMAL MODEL

We present two models in this section. The first is based on propositional logic to formalise role activation conditions, see section 4.1. It covers most of the ideas introduced in the previous sections, including appointment. We show in particular how to express the membership rules associated with active roles, and explain how we enforce these rules using event channels.

OASIS roles and appointment certificates include parameters. Role activation rules can match parameters to ensure, for example, that logged-in users can only invoke mutator methods on objects that they own. In section 4.2 we outline the extensions required to handle parameterisation. The extended model is based on first-order predicate calculus, which allows the use of variables in expressions. Our models are not designed to be application-specific. Instead, they are capable of expressing a variety of security policies.

4.1 Basic Model

The model is built on top of six basic sets, described as follows:

- $\mathcal{U}$: set of all users
- $\mathcal{S}$: set of all services
- $\mathcal{N}$: set of all role names
- $\mathcal{E}$: set of all environmental constraints
- $\mathcal{O}$: set of all objects
- $\mathcal{A}$: set of all access modes for objects

In addition to these sets, which are fundamental, two other sets are central to the interpretation of the basic model:

- $\mathcal{R}$: set of all roles
- $\mathcal{P}$: set of all privileges

A user is a human-being interacting with a computer system. An element in $\mathcal{U}$ can be any convenient representation that uniquely identifies a user in a system. The computer system is composed of a collection of services $\mathcal{S}$, which may be managed independently. A role is a named job function or title within an organisation that is associated with some service; a role is specific to a service, and is defined below. Services confer privileges on their role members, and may also recognise the roles of other services.

Definition 1. A role $r \in \mathcal{R}$ is a pair $(s, n) \in \mathcal{S} \times \mathcal{N}$, where $s \in \mathcal{S}$ is a service and $n \in \mathcal{N}$ is the name of a role defined by s.

The name of a role is unique within the scope of its defining service. When describing our model, we blur the distinction between roles and role names where this will not lead to confusion.

An environmental constraint $e \in \mathcal{E}$ is a proposition that is evaluated at the time of role activation. The value may depend on factors such as the time of day, the identity of the computer on which the current process is running or a condition such as group membership which requires access to a local database. In this paper we do not discuss the details of environmental constraints. We therefore consider each environmental constraint as an atomic proposition.

The conditions of some *role activation rule* must be satisfied when a role is activated. We may require in addition that some subset of these conditions, the *membership rule*, remains true throughout the session. If an environmental constraint e appears as a membership condition then its implementation must be *active*; when the role is activated each membership condition is evaluated, and in addition a trigger is set to notify the service should the condition become false. We discuss this requirement in more detail below.

A privilege is a right to perform some operation on a particular object. It is defined formally as follows.

Definition 2. A privilege $p \in \mathcal{P}$ is a pair $(o, a) \in \mathcal{O} \times \mathcal{A}$, where $o \in \mathcal{O}$ is an object and $a \in \mathcal{A}$ is an access mode for the object o.

The set of objects and their corresponding access modes are service dependent. For example, in relational database applications, objects may represent rows and their associated access modes include read- or update-attributes. In object-oriented systems, including distributed object systems, objects are represented naturally while access modes are the methods for each object. In general, we treat privileges as an abstract unit if the context permits.

The underlying idea of RBAC is to associate privileges with roles, and roles with users. These associations are described as relations in our model. Before describing these relations, we need to define the relationship between roles. As explained earlier, role hierarchy does not have any place in our model. Instead we control the acquisition of privileges through role activation governed by rules. Roles can only be activated during a session, and being active in one role may be a precondition for activating another; an example is a log-in credential that ensures that the user has been authenticated. In order to activate certain roles a user must hold an appointment; the corresponding condition in a role activation rule is an *appointment certificate.* Its formal definition follows.

Definition 3. An *appointment certificate* ω is an instance of an appointment. It may include a set of prerequisite roles, described by the function $\sigma : \Omega \rightarrow 2^{\mathcal{R}}$, where Ω is the set of all appointment certificates in the system.

An appointment certificate held by a user is valid only if the user is active in all of its prerequisite roles. This allows an appointer to ensure that an appointment certificate can only be used when the preconditions for activating all of those roles have been met.

A role activation rule specifies the conditions for a user to activate a role. It can be formally defined as follows.

Definition 4. A *role activation rule*, or activation rule for short, is defined as a sequent $(x_1, x_2, ..., x_n \vdash r)$, where x_i for $1 \leq i \leq n$ is a variable in the universe $X = \mathcal{R} \cup \Omega \cup \mathcal{E}$, and $r \in \mathcal{R}$. We say that each x_i for $1 \leq i \leq n$ is an activating condition for the role r.

For an activation rule $(x_1, x_2, ..., x_n \vdash r)$, a user must *satisfy* all conditions $x_1, x_2, ..., x_n$ in order to activate the role r. *Satisfaction* interprets each variable within the current context to give a Boolean value, see definition 6. There may be more than one activation rule associated with a particular role r.

An example of a role activation rule is given below with $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$ and $\Omega = \{\omega_1, \omega_2\}$, where $\sigma(\omega_1) = (\{r_3\})$.

$$r_1,\ \omega_1\ \vdash\ r_4$$

According to this rule a user who is active in role r_1 and holds the appointment certificate ω_1 can activate the role r_4, provided that the conditions for the appointment certificate to be valid are satisfied. In this case the sole condition is that the user be active also in the prerequisite role r_3.

This definition of activation rule is essentially a restricted form of Boolean logic. Any Boolean expression without negation over the universe X can be translated into one or more activation rules by rewriting it into disjunctive normal form (DNF), and taking each implicant as an activation condition. For example, an expression in the same universe as the above example is shown below in Boolean logic syntax.

$$(r_1 \vee r_2) \wedge \omega_1\ \vdash\ r_4$$

This is translated to DNF,

$$(r_1 \wedge \omega_1) \vee (r_2 \wedge \omega_1)\ \vdash\ r_4$$

and this can be written in sequent notation shown below.

$$\begin{aligned} r_1,\ \omega_1\ &\vdash\ r_4 \\ r_2,\ \omega_1\ &\vdash\ r_4 \end{aligned}$$

The set of all activation rules specified in a system is denoted by Γ. We summarise the symbols representing additional sets of objects in our model here.

- Ω: set of all appointment certificates
- Γ: set of all role activation rules
- Λ: set of all membership rules

We now consider a special type of role activation rule, called initial rules.

Definition 5. A role activation rule $(x_1, x_2, ..., x_n \vdash r)$ in which for $1 \leq i \leq n$ the variable $x_i \in \Omega \cup \mathcal{E}$ is *initial.* The role r is said to be an initial role.

Initial rules provide a means to allow users to start a session by acquiring initial roles. A particular case is that of rules with no antecedent conditions, $\vdash r$. The activation of such an initial role depends on system policies and typically requires system-dependent mechanisms, for example, password authentication or challenge-response authentication. In general activation of an initial role may require an appointment certificate and be subject to environmental constraints. The set of all initial roles is denoted $\mathcal{IR} \subseteq \mathcal{R}$. We restrict explicit association between users and roles to initial roles. In order to activate any other role a user must satisfy the preconditions of some activation rule, including possession of one or more prerequisite roles. These preconditions can include appointments and environmental constraints as well as role membership.

Note that during a session a user accumulates privileges by activating a succession of roles. Starting from a set of initial roles, which become active following authentication, a number of roles may be entered according to specified rules. An acyclic directed graph structure is therefore established that exhibits the run-time dependency of each role on its preconditions. Superficially the structure is similar to a static role hierarchy, but there are important differences. First, the dependency structure is dynamic; there may be several activation rules for the same role. Second, any privileges acquired by entering a role in this way will usually not be shared with any prerequisite role; it is more likely that the new role is more specific, and has been activated on the basis of appointment, or perhaps following database look-up. Parameter values play an essential part in determining which particular users can acquire the privileges associated with more specific roles.

The activation of a non-initial role requires a user to satisfy each of the conditions of some activation rule for that role. We define what is required for elements of each of the sets $\mathcal{R}$, Ω and $\mathcal{E}$ to satisfy a precondition for role activation.

Definition 6. The interpretation function for a role activation rule is a truth assignment with type, $I : X \rightarrow \{\textbf{true}, \textbf{false}\}$. An interpretation function, I, with respect to a user $u \in \mathcal{U}$ is denoted as I_u, and is defined below:

$$I_u(x) = \begin{cases} \textbf{true} & \text{if } x \in \mathcal{R} \text{ and } u \text{ is active in role } x, \\ & \text{if } x \in \Omega,\ u \text{ possesses the appointment certificate } x \text{ and is active in all the prerequisite roles } r \in \sigma(x), \\ & \text{if } x \in \mathcal{E}, \text{ and the evaluation of } x \text{ yields } \textbf{true}. \\ \textbf{false} & \text{otherwise} \end{cases}$$

Note that the definition of activation rules does not include negation ($\neg$). We briefly consider the effect of allowing negation of each of the three types of role activation condition. First, environmental constraints $e \in \mathcal{E}$ are atomic by definition. Any discussion of negation must take place in the context of an explicit environmental sublanguage, such as temporal expressions that test the time of day.

Second, appointment certificates $\omega \in \Omega$ record the fact of an appointment. It is only when a role is activated on the basis of an appointment certificate that any privileges are bestowed on the user. Negation should be associated with the roles activated rather than with the appointment certificate itself. In any case, appointment certificates can be anonymous and therefore transferable from user to user; the advantage of such a scheme is that a single revocation credential covers all the users of such an appointment certificate. It is not in general possible to tell whether a user has obtained such an appointment.

Negating a role $r \in \mathcal{R}$ makes perfectly good sense. Indeed, allowing a negated role among the conditions for role activation has a natural interpretation under I_u, namely $I_u(\neg r) = \textbf{true}$ if u is NOT active in role r. This is a possible implementation of a separation of duties constraint. But if a user must not activate two roles simultaneously, then the activation rules for each role should indicate that this user must not be active in the other. A more appropriate way of specifying the requirement would be to declare an explicit separation of duties constraint. We are actively investigating this issue.

Given the definition of the interpretation function, we can then define role activation formally.

Definition 7. (Role activation) A user $u \in \mathcal{U}$ can activate a role $r \in \mathcal{R}$ if and only if there exists an activation rule $(x_1, x_2, ..., x_n \vdash r) \in \Gamma$ such that $I_u \models x_i$ for all $1 \leq i \leq n$, where I_u is the interpretation function for u at the time when the activation request is made.

Note that the definition of the interpretation function implies that its evaluation with respect to a user changes with the context. When a user requests an activation of a role, the interpretation function is immediately evaluated in the user's context and the decision is made.

The opposite of role activation is role deactivation. Often continuing activation of a role will be valid only if some subset of the activation conditions continues to hold. These are called the membership conditions. The *membership rule* associated with a role activation rule specifies those conditions that must remain true in order for a user to remain active in that role.

When a role is activated at a service $s \in \mathcal{S}$ each of the conditions of the activation rule is verified. For roles associated with s itself this is straightforward. Roles and appointment certificates of other services must be validated by the issuer. In the case that x_i is a membership condition s establishes an event channel on the trigger $\neg x_i$ so that the issuer can notify s should the condition become false. OASIS depends on asynchronous notification to support role deactivation, see for example [3].

We have not defined explicit sublanguages for environmental contraints in this paper. However, it is worth considering two examples. First, let's suppose that a particular role may be held only between 1600 and 1800 hours on any day. We can include this requirement as part of the activation rules through a constraint in $\mathcal{E}$; at activation we check the time of day, say 1723, and set a timer exception for 1800 hours. In this instance the evaluation is independent of the user u.

Second, suppose that user u requests a privilege that is restricted to members of group g. For this example we require active database support. At activation, we check the data base for the required group membership. At the same time, we set a trigger for the negation of the condition. If the group manager updates the database to exclude user u then the trigger fires and deactivation takes place. This example shows how constraints in $\mathcal{E}$ may be user specific. The first prototype implementation of OASIS included a simple associative tuple store with triggers.

Definition 8. The *membership rule* associated with the activation rule $(x_1, x_2, ..., x_n \vdash r) \in \Gamma$ for the role $r \in \mathcal{R}$ is the sequent $(x_1, x_2, ..., x_m \vdash r)$ for some $m \leq n$, where x_i for which $1 \leq i \leq m$ are the membership conditions.

If a user is active in the role r through the activating rule $\gamma \in \Gamma$ for r, r shall be immediately deactivated if the associated membership rule $(x_1, x_2, ..., x_m \vdash r)$ can no longer be satisfied. We denote the set of all membership rules in a system as Λ. The formal definition of role deactivation is given below.

Definition 9. (Role deactivation) A role $r \in \mathcal{R}$ held by a user $u \in \mathcal{U}$ is deactivated if $I_u \not\models x_i$, where x_i is some membership condition in the rule $(x_1, x_2, ..., x_m \vdash r) \in \Lambda$ corresponding to the rule $\gamma \in \Gamma$ that activated r, and I_u is the interpretation function for u.

Continued satisfaction of the membership rule associated with the rule used for activating a role r is required for the user to remain active in r. Note that the deactivation of a role r may trigger the deactivation of another role r' whose membership depends on the membership of r. This is referred to as *cascading deactivation.* Its implementation is discussed in [3, 11] where the implementation of triggers and an event infrastructure are discussed.

Note that the membership rule associated with an active role r is specific to the rule under which r was activated. Consider as an example the rules obtained by translating the Boolean expression introduced after definition 4

$$(r_1 \vee r_2) \wedge \omega_1 \ \vdash \ r_4$$

which specifies that a user who is active *either* in role r_1 *or* in role r_2 and who holds an appointment certificate ω_1 may activate role r_4.

The corresponding activation rules are as follows:

$$r_1,\ \omega_1 \ \vdash \ r_4$$
$$r_2,\ \omega_1 \ \vdash \ r_4$$

In each case the membership rule will include the relevant prerequisite role, so as to enforce cascading deactivation at the end of the session. If revocation of the appointment is to take immediate effect, then the appointment certificate must also be a membership condition.

We can now define the association of roles with privileges. This is expressed as a relation as follows.

- $\mathrm{RP} \subseteq \mathcal{R} \times \mathcal{P}$, the role-privilege relation.

RP describes the role-privilege relationship. It is a many-to-many relation specified by the security administrators of an organisation to express security policies. We distinguish two sets of privileges for a role by the terms *direct* and *effective.* Our definitions are different from those given in [17], where direct and effective privileges are defined with role hierarchy in mind.

The *direct privilege set* of a role $r \in \mathcal{R}$ is the set of privileges assigned to r directly, i.e. $\mathrm{DP}(r) = \{p \mid (r, p) \in \mathrm{RP}\}$. The *effective privilege set* of a role r is the set of privileges that a user who is active in r must necessarily hold. This includes the effective privileges of all roles specified as membership conditions when r was activated, including the prerequisite roles of any appointment certificates. Each of these roles must still be active, or r would have been subject to cascading deactivation. The effective privilege set is dynamic, and depends on the specific activation history. The following definition ascends the activation tree recursively.

Definition 10. Suppose a user $u \in \mathcal{U}$ is active in some role r whose current membership rule is $(x_1, x_2, ..., x_m \vdash r)$. The effective privilege set $\mathrm{EP}(r)$ of r is defined as follows:

$$\mathrm{DP}(r) \cup \bigcup_{1 \leq i \leq m} \begin{cases} \mathrm{EP}(x_i) & \text{if } x_i \in \mathcal{R} \text{ is a prerequisite role} \\ \mathrm{EP}(\rho) & \text{if } x_i \in \Omega \text{ is an appointment certificate, the union of for all prerequisite roles } \rho \in \sigma(x_i) \end{cases}$$

The effective privilege set for role r defines privileges that a user must continue to hold while remaining active in that role.

In some RBAC models it is possible to compute the maximum privileges that a user may assume. OASIS defines security policies on a service by service basis for multiple management domains in a distributed world. For example, a nationwide system for electronic health records will comprise many interoperating domains such as hospitals, primary care practices, clinics, research institutes etc. Services within a given domain express their policy for role activation and service use. Membership of a role of one service may be required as a credential for entering another. Such dependencies are specified in service level agreements. It is likely that policy will be administered at domain level, and will derive from local and national administrative and legal sources, depending on the application. Service level agreements will also be made across domains. Appointments may be made at several administrative levels. Some appointment certificates will apply to many domains, for example those representing academic and professional qualifications. Others will be dynamic and local, for example temporary substitution for a colleague who is called away while on duty.

Should it be required, it is possible to compute the maximum privileges that a user may obtain based on statically known appointments. This assumes that all constraints will be satisfied at the time roles are activated. In practice, dynamic environmental conditions may prevent some roles from being activated in any specific session. In addition unforeseen appointments might be made dynamically within sessions.

Previously we introduced *appointment certificates* to represent appointments. Services which support appointment will define their own roles and policies to manage it, and will issue and validate the appointment certificates. At each appointment an appointment certificate is returned to the appointer, who subsequently transfers it to the appointee. The latter can then use the appointment certificate during role activation, either at the issuing service or at some other. The role activation rules may specify a number of prerequisite roles in addition to one or more appointment certificates. In this way we can for example implement the two-signature, countersign approval system commonly found in business by requiring two appointment certificates. In addition the appointer, when applying for the appointment certificate, may specify a set of roles in which the appointee must be active in order for it to be valid, see definition 3.

OASIS supports rapid and selective revocation, which is managed by invalidating the appointment certificate issued on an appointment. Whenever a credential is invalidated any roles that depend on it are deactivated and their associated credentials invalidated, see [3] for implementation details. This can lead to cascading deactivation of a tree of roles in which a user is active. Cascading deactivation also takes place when a session ends after a user logs off; *logged-in-user* is likely to be an initial role. This basic model is sufficient to support system-managed revocation. We do not attempt to formalise the details.

4.2 Extended Model

There is an increasing interest in the research community in just-in-time, active security where policies must adapt to their environment. Prominent examples include the workflow authorisation model [1] and task-based authorisation controls (TBAC) [23]. Essentially, a major drawback of traditional RBAC models that limits their usefulness is their inability to take into account fine-grained information from the execution context. In the introduction we discussed adhoc extensions that have been suggested to meet specific needs. As an alternative we propose a generic framework which can be tailored to each application domain with minimal effort.

The problem of implementing an access control system for the Electronic Health Records (EHRs) of the United Kingdom (UK) National Health Service (NHS) is one of the case studies that has informed the design of OASIS [2]. In this application it is vital that the user requesting access can be identified, since under the UK Patients' Charter the patient has the right to exclude named individuals. Traditional functional roles are not adequate for this purpose, since such an exclusion is specific both to the patient and to the potential reader. Individual identity must therefore be established by a credential which is presented on access; in the OASIS model the obvious choice is a role membership certificate, asserting the NHS identifier of the user in some way. Rules for role activation refer to individual roles, so it is not possible to name a separate role for each potential user; such an implementation would be neither manageable nor scalable. Instead we set up generic initial roles such as *logged-in-user* and *smart-card* to correspond to the modes of authentication supported by the system. In order to identify the individual we extend the role membership certificate (RMC) by a parameter, *userid* and *NHS-id* respectively. Such parameters are among the fields protected when the RMC is generated, see for example [3].

In the basic model described in the last section access control decisions are made on the basis of propositions that are evaluated in the current context. These propositions relate to roles and appointments, and the policy governing the acquisition of privileges is expressed in terms of them. Role activation rules can take account of the execution environment by evaluating propositions relating to such factors as the current time of day or an entry in an administration database. We extend this model to allow parameterisation of roles, appointment certificates, privileges and environmental constraints. In order to accommodate these extensions we have enhanced the specification to define role activation rules and membership rules in terms of predicates rather than propositions.

The details of these extensions are intuitive. RMCs contain a number of protected fields, which, together with a nonce key identifying the session, form input to the one-way function that generates the signature. In the formal model a parameterised role consists of a role $r = (s, n)$ together with a k-tuple that identifies the parameter values. In expressing the conditions for role entry a proposition identifying r as a prerequisite role is replaced by a k-ary predicate. Within a rule the arguments of each predicate may be either variables or constants. Variables may also occur in the parameterised role r' that is being activated. Evaluation of an activation rule proceeds by unification over the variables that are specified. Values in the RMCs that establish membership of prerequisite roles set the corresponding variables appearing in activation conditions. Activation succeeds only if all conditions can be met, and subject to a consistent assignment of values to variables. In this case the parameters of the new role r' are set from the variable values established during unification.

Environmental constraints are in detail application specific, but a common and useful form is the ability to check information in a database during role activation. Provided the database itself can be identified then such a constraint can be viewed as a predicate assertion. For a relational database the natural interpretation is of the occurrence of a tuple in a relation named by the constraint. In a deductive database the predicate assertion specifies a query directly, and unification over the variables involved has a direct counterpart during query evaluation.

Parameterised appointment certificates are similar to parameterised roles. If an appointment certificate with k parameters appears as a precondition for role activation then the activation rule includes a k-ary predicate. Variable values are matched against other occurrences of the same named variable within the rule.

A parameterised appointment certificate is valid only if its holder is active in all of its prerequisite roles. In addition, parameters of credentials (RMCs) associated with these roles may be required to match parameters of the appointment certificate. An appointment certificate may also be subject to one or more environmental constraints. The relationship between a parameterised appointment certificate and its prerequisite roles is specified in a validity rule, defined as follows.

Definition 11. A validity rule for a parameterised appointment certificate ω is defined as a sequent $(x_1, x_2, ..., x_n \vdash \omega)$, where $x_i \in \mathcal{R} \cup \mathcal{E}$ for all $1 \leq i \leq n$.

Note that only parameterised roles or environmental constraints are allowed in the premises part of a validity rule. During role activation any appointment certificates are validated before the activation rule is evaluated. Unification of variables in the validity rule may constrain undefined parameters of the appointment certificate; the values set form part of the context during role activation.

The roles held by a user determine that user's privileges.

A privilege can typically be considered as a specific access right at a service. Roles are parameterised, and parameter values can be propagated to privileges at request time. For example, the privileges corresponding to a role $\mathit{writer}(\mathit{userid})$ activated at a file service may be restricted to files that are owned by the user named *userid*. We do not go into details here.

The first prototype implementation of OASIS used simple parameter matching at role activation time, essentially setting parameters when they were first encountered and denying role activation if there was a later conflict. Parameters were strings, and the only value comparison supported was equality. Database look-up was supported by an associative tuple store.

If an environmental constraint appears in a membership rule then the service which evaluates it must have an active implementation. One reason for using such a naive database was that it was easy to set up event channels for it; if a query during activation represented a membership condition then an event channel was established, and the role activation service could be notified if the condition became false subsequently. We have just started to experiment with the POSTGRES object-relational database management system (DBMS), which allows agents external to the DBMS to set triggers in order to receive notification of database update.

Initial experiments are encouraging, but a lot of work remains to be done. If this approach is successful then it would be natural to regard parameter k-tuples as instances of classes, and to enforce type checking of individual parameters. That would be an obvious improvement.

5. EXAMPLES

Suppose that privacy legislation has been passed whereby someone who has paid for medical insurance may take certain genetic tests anonymously. The insurance company's membership database contains the members' data; the genetic clinic has no access to this and the insurance company may not know the results of the genetic test, or even that it has taken place. The clinic, for accounting purposes, must ensure that the test is authorised under the scheme. A member of the scheme is issued a computer-readable membership card containing an appointment certificate and the expiry date. In the simplest scheme the membership card is authenticated at the clinic, the member enters the unparameterised role *paid_up_patient* and the test is carried out.

$$\vdash \mathit{paid\_up\_patient}$$

In the presentation above *paid_up_patient* is an unparameterised initial role with no explicit preconditions. Checking the expiry date on the membership card is part of the system authentication process. A more likely scenario is that the activation rule for *paid_up_patient* comprises the appointment certificate and an environmental constraint requiring that the date of the (start of the) treatment is before the expiry date of the insurance scheme membership. The appointment certificate is validated at the issuing service (a trusted third party) before role activation can proceed. In this case the appointment certificate becomes the membership rule for the role *paid_up_patient*; if the appointment certificate is found to be fraudulent treatment is terminated.

It is easy to express the constraint on the expiry date using parameters. Suppose that the expiry date is a protected field of the appointment certificate, and the environmental constraint ϵ_1 checks that a temporal argument lies in the future. The following activation rule matches the parameter t, reading it from the appointment certificate and supplying it as argument to ϵ_1.

$$\omega_1(t),\ \epsilon_1(t)\ \vdash\ \mathit{paid\_up\_patient}$$

An initial role *logged_in_user(uid, machine)* might be defined so that the user-id and the machine at which the login has taken place can be carried forward and checked as environmental conditions on subsequent role activation. Again, at the engineering level, the parameters are protected fields in the role membership certificates. If login can be at any computer in the administrative domain we define the role *logged_in_user(uid)* with a single parameter. In the health-care domain everyone has a unique NHS identifier which could be used as a parameter.

We now work through an example scenario from an A&E department of a hospital. Let us suppose that some of the roles involved are *nurse(x)*, *screening_nurse(x)*, *doctor(x)* and *treating_doctor(x,y)* where x is the identity of the role holder in each case and y is the patient being treated. In outline, as hospital staff come on duty they login and activate roles such as nurse, screening nurse and doctor. When someone goes off-duty they logout and the roles they hold are deactivated. As an example of how dynamic appointment might be used we suppose that a screening nurse assigns each patient that arrives to a particular doctor who becomes the treating doctor for that patient.

There is an electronic health record (EHR) service in the hospital domain which interacts with a National EHR service, external to the domain, in order to assemble any required, and authorised, records of treatments the patient has had. Let us assume that the general policy is that screening nurses may read patients' contact and emergency data only, and that treating doctors may read the EHR of any patient y they are treating while they are active in the role *treating_doctor(x,y)*. The EHR service recognizes the A&E service roles described above (a service-level agreement) and implements this policy.

For the key roles in this scenario we now define the activation rule and the membership rule. We also demonstrate long-term and dynamic appointment.

When a doctor or nurse is employed at the hospital their academic and professional credentials are checked and they are issued with an appointment certificate parametrised with their identity information. This might be on a computer readable card or be stored in the administration filespace. For the role *doctor(x)* the activation rule comprises x's appointment certificate. The membership rule is identical to the activation rule since the appointment certificate must remain valid for x to remain active in the role.

$$\omega_2(x) \vdash \mathit{doctor}(x)$$

The role *nurse(x)* has a similar structure.

For the role *screening_nurse(x)* the activation rule comprises the prerequisite role *nurse(x)* and has no appointment or environmental conditions. The membership rule is identical to the activation rule.

$$\mathit{nurse}(x) \vdash \mathit{screening\_nurse}(x)$$

For the role *treating_doctor(x,y)* the activation rule comprises the prerequisite role *doctor(x)* and there is no environmental condition. An appointment certificate is required. This is a certificate allocated by the screening nurse to doctor x authorising her to treat patient y.

$$doctor(x),\ \omega_3(x,y)\ \vdash\ treating\_doctor(x,y)$$

The role *doctor(x)* could also be a prerequisite role of the appointment certificate, but it is redundant in this example. Note that if the screening nurse goes off duty and logs out, she deactivates her role *nurse(x)* causing the dependent role *screening_nurse(x)* to be deactivated. The appointment certificate $\omega_3(x,y)$ is not invalidated at the end of her session. The membership rule of the role *treating_doctor(x,y)* is once again identical to the activation rule. First, x must remain active in the prerequisite role *doctor(x)*, in order to ensure that the role *treating_doctor(x,y)* is deactivated at the end of the session. There are two reasons for making $\omega_3(x,y)$ a membership condition as well. The screening nurse may wish to reassign the patient to another doctor, and in any case the appointment should be revoked when patient y is discharged. Note that x may still be on duty when this happens.

A given doctor will be assigned to a number of different patients while on duty and will activate the role *treating_doctor(x,y)* for each of them. The role *treating_doctor(x,y)* of the A&E service gives doctor x the privilege to access patient y's health record at the EHR service. Other hospital services such as the Pharmacy service and the X-ray service will also be OASIS-aware and require the A&E role membership certificate *treating_doctor(x,y)* on invocation. Such services will record the parameters x and y for accounting and audit.

6. CONCLUSION

OASIS is an access control system for open, interworking services in a distributed environment modelled as domains of services. Services may be developed independently but service level agreements allow their secure interoperation. OASIS is closely integrated with an active, event-based middleware infrastructure. In this way we continuously monitor applications within their environment, ensuring that security policy is satisfied at all times. We therefore address the needs of distributed applications that require active security. Any formalisation must take account of the relationship between OASIS and the underlying active platform.

In this paper we have formalised the OASIS model without reference to domains. Formal specification is crucial in order to manage access control policy for future, large scale, widely distributed, multi-domain systems. A formal model allows policy components established across a number of domains to be checked for consistency. This is necessary, since otherwise policy cannot be deployed by domains acting autonomously; for example, a government edict might require changes of policy across heterogeneous healthcare domains. Automation is essential to minimise human error, and it can only be used where a formal model exists.

OASIS is role based: services name their client roles and enforce policy for role activation and service invocation, expressed in terms of their own and other services' roles. A signed role membership certificate is returned to the user on successful role activation and this may be used as a credential for activating other roles, according to policy.

We do not use role delegation. Instead, we have defined *appointment* which we believe to be both more intuitive and more applicable in practice. Appointments may be long-lived, such as academic and professional qualification, or transient, such as standing in for a colleague who is called away while on duty. On appointment, the appointee is issued with an appointment certificate which may be used, together with any other credentials required by policy, to activate one or more roles.

In addition to role membership certificates and appointment certificates role activation rules may include environmental constraints. Examples are user-independent constraints such as time of day and conditions on user-dependent parameters. For example, it may be necessary to perform database lookup at a service to ascertain that the user is a member of some group. Alternatively, a simple parameter check may ascertain that the user is a specified exception to a general category who may activate the role.

The membership rule for a role indicates those security predicates for activating the role that must remain true for the role to continue to be held. Event channels are set up between services to ensure that all conditions of the membership rule remain true. Should any condition become false this triggers an event notification to the role-issuing service and the role is deactivated for that user. By this means we maintain an active security environment.

OASIS is session based. Starting from initial roles, such as "authenticated, logged-in user", a user may activate a number of roles by submitting the credentials required to satisfy an activation rule. The activated roles therefore form trees dependent on initial roles. Should any membership condition for any role become false the dependent subtree is collapsed. If a single initial role is deactivated (the user logs out), all the active roles collapse and the session terminates.

Our application domains require parameterisation of roles. For example, in the healthcare domain a patient might specify "all doctors except my uncle Fred Smith may read my health record". For a filing system it is necessary to indicate individual owners of files as well as groups of users.

Future work involves the detailed modelling of role parameters. In practice, distributed systems comprise many domains; for example the healthcare domain comprises hospitals, primary care practices, research institutes etc. We will generalise our naming structure to include domains explicitly. We are working on the management of policy for role activation and service use. Policy may derive from local and national sources and will change over time. Policy stores will be managed using OASIS in our active environment. The formalisation of OASIS will provide a firm basis for requirements such as checking the consistency of policies.

7. ACKNOWLEDGEMENT

We acknowledge the support of the Engineering and Physical Sciences Research Council (EPSRC) under he grant *OASIS Access Control: Implementation and Evaluation.* Members of the Opera research group in the Computer Laboratory made helpful comments on earlier drafts of this paper. The ideas that lie behind definition 11 were introduced by John Hine, see [12]. We are grateful to Jon Tidswell and the anonymous referees for constructive criticism which has much improved this paper.

8. REFERENCES

[1] V. Atluri and W.-K. Huang. An authorization model for workflows. In *4th European Symposium on Research in Computer Security*, pages 44–64, Rome, Italy, September 1996.

[2] J. Bacon, M. Lloyd, and K. Moody. Translating role-based access control policy within context. In *Policy 2001, Workshop on Policies for Distributed Systems and Networks*, Bristol, UK, Jaunary 2001.

[3] J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, and M. Spiteri. Generic support for distributed applications. *IEEE Computer*, pages 68–76, March 2000.

[4] E. Barka and R. Sandhu. Framework for role-based delegation models. In *16th Annual Computer Security Applications Conference*, New Orleans, Louisiana, December 2000.

[5] E. Barka and R. Sandhu. A role-based delegation model and some extensions. In *23rd National Information Systems Security Conference*, Baltimore, MD, October 2000.

[6] E. Bertino, E. Ferrari, and V. Alturi. A flexible model for the specification and enforcement of authorization constraints in workflow management system. In *Second ACM Workshop on Role-Based Access Control*, pages 1–12, Fairfax, Virginia, November 1997.

[7] M. J. Covington, M. J. Moyer, and M. Ahamad. Generalized role-based access control for securing future applications. In *23rd National Information Systems Security Conference*, Baltimore, MD, October 2000.

[8] D. F. Ferraiolo, J. F. Barkley, and D. R. Kuhn. A role-based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information and System Security*, 2(1):34–64, Feb 1999.

[9] L. Giuri and P. Iglio. Role templates for content-based access control. In *Second ACM Workshop on Role-Based Access Control*, pages 153–159, Fairfax, Virginia, November 1997.

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

[11] R. Hayton, J. Bacon, and K. Moody. OASIS: Access Control in an Open, Distributed Environment. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA, May 1998. IEEE.

[12] J. Hine, W. Yao, J. Bacon, and K. Moody. An architecture for distributed OASIS services. In *Middleware 2000*, pages 104–120, New York, NY, April 2000.

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

[14] J. D. Moffett. Control principles and access right inheritance through role hierarchies. In *Third ACM Workshop on Role-Based Access Control*, pages 63–69, Fairfax, Virginia, October 1998.

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

[16] M. Nyanchama and S. Osborn. Access rights administration in role-based security systems. In J. Biskup, M. Morgernstern, and C. Landwehr, editors, *Database Security VIII: Status and Prospects*, 1995.

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

[18] J. Saltzer and M. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, September 1975.

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

[20] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-Based Access Control Models. *Computer*, 29(2):38–47, Feb. 1996.

[21] R. T. Simon and M. E. Zurko. Separation of duty in role-based environments. In *10th IEEE Computer Security Foundations Workshop*, pages 183–194, Rockport, Massachusetts, June 1997.

[22] R. K. Thomas. Team-based access control (TMAC): A primitive for applying role-based access controls in collaborative environments. In *Second ACM Workshop on Role-Based Access Control*, pages 13–19, Fairfax, Virginia, November 1997.

[23] R. K. Thomas and R. S. Sandhu. Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented authorization management. In *IFIP WG 11.3 Workshop on Database Security*, Lake Tahoe, California, August 1997.

[24] W. Wang. Team-and-role-based organizational context and access control for cooperative hypermedia environments. In *ACM Hypertext'99*, February 1999.