Towards a more Complete Model of Role

Cheh Goh, cng@hpl.hp.com
Adrian Baldwin, ajb@hpl.hp.com

Extended Enterprise Lab,
Hewlett Packard Laboratories,
Stoke Gifford, Bristol BS34 8QZ
United Kingdom

Abstract: *In order to manage the use of roles for the purpose of access control, it is important to look at attributes beyond the consideration of capability assignment. Fundamentally, a generic attribute description using a constraint-based approach will allow many of the important aspects of role, such as scope, activation and deactivation, to be included. Furthermore, the commonly accepted concept of role hierarchy is challenged from the point of view of subsidiarity in real organisations, with the suggestion that role hierarchy has limited usefulness that does not seem to apply widely.*

Keywords: *Role-based, access control, RBAC, management, constraint-based.*

1 Introduction

Recent interest in role-based access control (RBAC) has grown from attempts to understand its relationship with other types of access control such as the traditional DAC and the military grade MAC, to the management of roles with respect to users and capabilities. The definitions of roles are varied, but are mostly associated with a set of responsibility-capability pairs. Closer examination suggests that in a complex environment it is better to keep role related aspects as role attributes rather than expressing them as policies outside the scope of the role administrator. These extra attributes provide a more flexible means of specifying access control in a wide range of situations.

More specifically, many of these attributes are useful for the management of the roles, and allow a more powerful means of ensuring flexible and effective access control to many types of objects. This paper aims to highlight aspects of role which has so far been ignored in most of the discussion of RBAC and yet, will be important when we extend RBAC to the internet domain.

These days, computers are supporting a larger number of interactions, both formal and informal, occurring between ever more diverse and distributed teams (even spread over several companies), probably using the internet. A software development project, for example, will have a number of formal interactions such as design approval, test specification, signing off of deliverables and milestones, as well as the less formal ones such as swapping bits of code and exchange of ideas.

These tasks would be supported by a number of applications such as CASE, source control, design packages, as well as general project management, workflow and people tracking tools. Less formal interactions would be supported by e-mail, video conferencing and discussion groups. Underlying the rich set of interactions between people, applications and information there needs to be a powerful security system and, therefore, security management system, especially if these functions are performed over the (relatively insecure) internet.

People who are contributing to such a project obviously have particular roles and responsibilities from which many of the security obligations and constraints can be derived. In these types of situations a person can play a number of different types of roles which are discussed in the next section. For example a software team leader will play a line management role for their team members as well as managing the teams' output. This paper argues that a richer role model is required to support the various types of role and to ease the management of fine grained access control needed for a range of different types of application and data. This should be achieved through tying the business level interactions to the underlying system functionality.

We propose the use of additional role attributes specified using logical constraints to enrich the role model allow-

3rd ACM Workshop on Role-Based Access Fairfax VA

ing the full range of roles to be expressed. The use of constraint-based descriptions for a number of role attributes, draws heavily on the idea of applying constraint-based descriptions to policy as discussed in [Goh97]. The expanded role model brings role-related issues that could be expressed as policies into role thereby helping to separate the administration of role from that of policy. The enhanced attributes should enable roles to be more easily applied to the wide range of situations which would be encountered when using the full range of business and technical computing services.

Questions are also raised to challenge the commonly accepted concept of role hierarchy with respect to subsidiarity in section 4. This is done with the aim of exposing the need to balance practical role-based concepts with interesting theoretical exposition, to achieve a better understanding of role and its significance in the access control area.

2 Types of Role

In general, roles can be considered from several different angles: organisation, relative relationship, grouping of convenience and grouping through selection. This section reviews these uses of role looking at different aspects and features which may be required for each different view of a role. These aspects and features can be expressed using constraint-based role attributes such as those proposed in the next section. Throughout this section roles associated with a software development project, Figure 1, are used to illustrate various concepts.

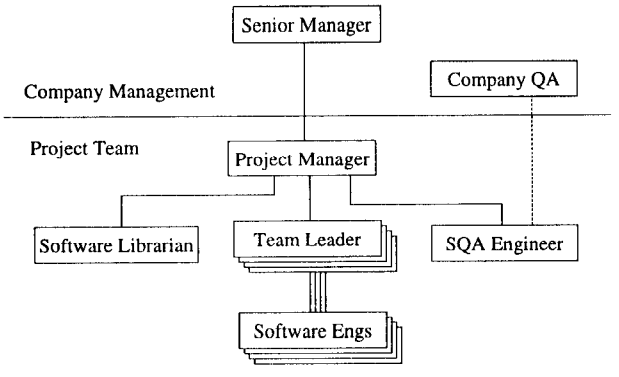

Figure 1: Example organisation for a software project.

In considering these types of roles we note that role should serve as more than just an access limitation mechanism. Roles are defined to represent various job functions within an organisation and, as such, can be used for business level policies as well as for access control. At the level of business applications the line between the two becomes blurred. Business policies may state that a project manager must approve expense claims for that project, but the access control on the expense claims system would limit approval to the project manager's role within the scope of their project.

2.1 Organisational role

Organisational role has been created as a way to assign responsibility and capability within organisational units. An organisation provides the environment within which a role is meaningful. Suppose we regard role as object, then its environment of validity—sometimes also known as the *domain* in which the role is meaningful—can be considered as an object instance variable [Lupu97]. Accompanying an organisational role is the responsibility of a person with that role towards that *environment*; and therefore the *scope* of the role is an essential feature.

Encapsulation decisions can be used to control which roles are exported outside the immediate environment. The role of a research engineer, for example, is usually hidden from the customers' view of the environment called a "company". Customers are, generally, familiar with the "marketing manager", the "sales representative" and the "customer support engineer". In these terms the scope attribute can be used to make decisions about the access control for communications between roles.

Having defined organisational roles based on responsibilities within a certain scope other attributes can be important in the management of these roles. For example, a delegation attribute is important when considering how to manage the role when the role player is absent. Certain organisational roles will have an activation attribute defining at what point the role becomes enabled. For example, there would probably be an auditor role within the software project. This role would have comprehensive rights to read all documents but only whilst the activation attribute of 'audit happening' is true.

It is important to note that the organisation diagram defines reporting relationships and is not a role hierarchy; role capabilities are assigned according to the processes performed within a particular environment.

2.2 Object specific role

When the responsibility and capability of a subject interacts with that of another, there is the concept of role interaction. This has been amply discussed in [Lupu98], under both the concept of role relationships and relation-

ship classes. Apart from this significant concept, there is a need to recognise the relative nature of roles with respect to the role player. For example, the concept of "manager" may often be understood in two ways: the manager of a project, and the manager of specific objects within that project. In the software project organisation the team leader will have an organisational role as a manager of a particular work package and a object specific role relative to the staff in the project. This object specific role is meaningful in the same environment, but it is a lot more specific because it relates to another person, or persons with a certain role. Each team leader may have the same personnel management capabilities but these would be relative to the staff they manage. In this case a role's capabilities are dependant on other objects (eg team members) and the role and privileges could be parameterised [Giuri97]. An alternative to a parameterised role name is to have a parameterised scope attribute enabling access for objects meeting the attribute's constraint.

Some of the team's software engineers could be contractors and the team leader would not perform this relative person management role whilst they will still hold the organisational role. In this case a parameterised scope constraint can be used to describe when this role is valid for particular objects (ie which staff each team leader manages).

Modern organisations that try to be flexible in order to optimise their operation often employ what is known as a lattice structure or matrix management, whereby a specialist may have a direct "line" manager deciding her salary, a "reporting" manager in whose project she works, and a "technical" manager to whom technical matters are reported. In this case, object specific role provides distinct views according to the role player, and adds clarity to the notion of role, something that eludes the definitions given in [Lupu98].

The single most important application of this concept is in enforcing RBAC related to policies that reflect personal data privacy laws, where role player's identity must be taken into account. For example, a constraint saying who is being managed for any instance of a line manager would easily allow personal information to be restricted to each persons line manager.

2.3 Grouping as role

Often roles are created simply for ease of reference. "Company employee" is a simple way of referring to a group of people and the associated responsibilities and privileges. All company employees would enjoy all the capabilities of this role. This is in addition to their specific job related capabilities derived from their primary role representing their capability and responsibility within the organisation. Equally if a team has both contractors and company employees as software engineers, both may have equal capabilities as software engineers within the team but each will have different capabilities according to their employment status.

The validity of regarding grouping of convenience as role is due to the experience of practically dealing with organisations in which set privileges can be spoken of and manipulated flexibly. In this case it is useful to give an individual both their organisational role defining the specific job capabilities as well as role groupings such as 'employee' or 'contractor'. The use of grouping in this way ensures that attributes concentrate on the job specific capabilities and are not confused by peripheral issues such as employment status.

2.4 Unspecific role

Role has its associated capabilities and responsibilities. The relationship between each pair of capability and responsibility is often static and appears as a direct mapping. Capabilities are assigned to a role when that role is defined and clearly different roles may share a number of common capabilities. Also, it is possible to establish a role that requires specific *qualifications* and *activation* criteria (see section 3.3 & 3.4).

The unspecific role is more dynamic in that it is formed because a role holder possesses the correct attributes and capabilities for a particular situation; for example, *'access will be given for any role that has capability $\mathcal{C}$'*. One commonly used example is in a workflow environment: a task can be performed at a particular stage by, not so much the pre-specified role, but by anybody with a role that has some qualifier. The qualifier is usually a set of constraints such as "the ability to approve a cheque for up to $2000.00". When a qualifier is used for access control instead of a role, role plays a secondary part in the access control mechanism, very much like how user ID is relative to RBAC.

Clearly the identity of user and the role that is active at the moment of permitting or denying access are both important for audit trail purposes. Nonetheless, the approach of capability based control points out the way RBAC can be extended practically.

3 Role Attributes

The attributes associated with a role has variously been discussed in [Chen96, Lupu98]. A more sophisticated

approach than that taken in [Chen96] is needed as soon as the elementary model is to be extended. The attributes highlighted below, lack of explicit mention so far in the literature, are included mainly for their usefulness.

Why do we need more sophisticated role attributes? In the internet-manic world, RBAC is finding its usefulness extended far beyond the world of database security, which has helped demonstrate the needs of different access control paradigms. Once a well-understood environment ceases to be guaranteed, in the world of information, it is necessary to create common semantics for different organisations, entities and domains to interact meaningfully. As much work has gone into standardising information and system representation such as DMTF's Common Information Model (CIM) [DMTF-CIM], it is practical to seek role definition commonality by specifying role attributes through standard descriptions, rather than attempting a standardisation of role definitions.

Attention is drawn to the similarity between the concept raised in the RBAC models in [Sandhu96] and the description in the following subsections. The distinction is vital, in that the concept of "constraints" in [Sandhu96] applies to the RBAC components and the relationship between different components, and may in principle be summarised simply as a set of policies [Goh97]. Role attributes are, however, meant to describe the inherent aspects of a role without explicit reference to other objects. The fact that constraints will be mentioned as a convenient way to describe attribute is coincidental and is really an extension of an earlier concept in [Lawrence93]. Role attributes are, therefore, orthogonal to, but interact with, the "constraints" governing RBAC in [Sandhu96].

3.1 Scope

The consideration of role is often done within an assumed domain. This is convenient in making the principles of RBAC clear and in making comparisons with other types of access control such as DAC and MAC. The assumption of a pre-set domain should be removed when we begin to consider roles in a more general way. A burgeoning idea along this line can be found in [Moffett94]. The approach taken in [Lupu97] clearly offers a solution to dealing with the reality that, for example, the role of a nurse in hospital X is not the same as that in hospital Y, and the role of a nurse in ward A is not the same as that in ward B. The instantiation of the "nurse role" object allows the instant variable representing the scope to be defined where necessary.

An extension of this object oriented approach is needed when we consider dynamic role instantiation and assignment. When can an instance of a role be established? Can a role be assigned to a user in a given environment whilst constrained by the RBAC policy? The answers to these questions become the scope of a role. The answers themselves can be established through the use of logical constraints as part of the role attribute. A relevant set of constraints as part of a role will enable the role manager to determine the appropriateness of the application domain and potential automation of role to user assignment. For example, when creating a software project the manager will look for people with a software engineering role who can therefore be assigned to a particular role in the project. That is a scope attribute can be defined using logical constraints to describe the valid projects in which a software engineer is entitled to operate. This is a generic approach similar to that taken for the description of policy [Goh97] and of RBAC model [Sandhu96]. Similar ideas can be expressed using policies outside of the roles and the decision of whether to keep information within a role as an attribute or outside a role as a policy is not obvious and will often depend the preferred way of administering these concepts.

3.2 Activation Criteria

Once a role is established, it can be made to be intrinsically relevant only under certain conditions within the pre-defined scope. These conditions constitute the constraits that are separate from the RBAC policy constraints. Apart from the "where", "when" and "what" mentioned in [Lawrence93], the most common additional activation considerations include:

- **Event-triggered condition.** The role of "fire marshal" would only be activated when there is a fire alarm in a building. From this point of view, the trigger must be well defined, otherwise it will not be possible to have the correct level of access control because the role is usually not activated. Clearly it is possible to include time-based condition as an event trigger, but this separation provides a convenient view mainly for ease of management.
- **Composite condition.** The constraints that need to be satisfied in order for a role to become active are not always limited to event triggers. An event is at best a trigger, and if the state of the system is such that a constraint is satisfied, then a role becomes active. This is very similar to role scope

establishment which is orthogonal to the activation requirements.

In the literature, role has been regarded as a collection of policies in [Sloman94]. This treatment will subsume the requirements mentioned in this section, provided that the way policies are described is sufficiently generic, with descriptive power at least as encompassing as that proposed in [Goh97].

3.3 De-activation Criteria

For how long would a role be valid from the time of its activation? Roles may remain active for a pre-set period of time, indicated as a role attribute, or it may also become inactive based on certain other constraints. In the simplest cases, a time-to-live (TTL) specified in terms of duration or event count is sufficient. More generally a full constraint-based approach, in a fashion similar to role activation, would be needed.

3.4 Qualification

Another dimension in the attributes of role is qualification. When can a person be considered for taking up the role of a security officer? The absence of criminal record could be one of the fundamental requirements. Another qualification may be that the person has relevant experience for more than a number of years. In the software project example the capability of accepting test results may only be performed by an qualified test engineer, and such conditions may even be a contractual requirement. The fulfilment of a role requires the satisfactory discharge of the associated responsibility as well as the access right. A computer playing the role of access controller must have sufficient memory, computing power, and the appropriate software to effect quick control, which may rule out, say, an Intel 286 machine that survived from one's younger days!

The advantage of having qualification as an attribute is the potential to ease or automate role assignment. The process of selecting the qualified candidate as potential owner of a role is greatly simplified, allowing the role assignment manager to make decisions quickly.

3.5 Delegation and Transfer

One of the most interesting features of role is the possibility of delegating or transferring its associated capability. The distinction between delegation and transfer made in the context of database access control [Bertiino97] is, clearly, equally applicable for roles, such that when a role is delegated from one person to another, the delegator retains the capabilities, whereas if a role is transferred from one person to another, the transferor will no longer retain the capabilities. How delegable and transferable a role is could be an attribute of the role itself. Previous discussion on this area can be found in [Yialelis96] which looks at relatively simple scenarios.

For convenience, we coin the phrase "role empowerment" to mean either role delegation or role transfer. A more complete approach to role empowerment necessarily include, but not limited to, the following considerations:

- **Empowerment qualification.** In the same way that a role should only be assigned to a person with the necessary qualification, the empowerment of capabilities to a delegatee or transferee must be conditioned upon the possession of certain qualification. The role of a nuclear power station operator, for example, cannot be delegated most of the time because there are not many people trained to shut down a nuclear boiler!
- **Empowerment conditions.** When may a role be delegated and when may it be transferred, if role empowerment is permitted? In addition to empowerment recipient's qualification, there could be conditions for the empowerment to take place, such as when the existing role owner is no longer fit to discharge a responsibility. Again, a constraints-based approach would potentially capture all the necessary information to make a decision.
- **Delegability and transferability set.** What are the capabilities that can be delegated or transferred, and under what condition?
- **Degree of delegability.** This is about the number of times a role can be assigned to someone else by the present owner of that role. If the degree is zero, it can be used as an indication that the role should never be handed over to another person by the owner, while degree of one can be used to represent a single delegation or transfer and no more, as in the case of "signatory power holder" in many organisations. Infinite degree of delegability or transferability implies that a role can be handed from one owner to another without any limit in terms of counts.

- **Degree of transferability.** This is similar to delegation, except that the transfer will lead to the original role owner to relinquish all capabilities and responsibility.
- **Delegation and transfer relationship.** Can a role delegated to a user be transferred, or can a transferred role be delegated? Can the original owner of a transferred role be the recipient of a delegation or transfer of the same role? How can the constraints governing these two be accurately and adequately described?
- **Delegation/transfer count.** How many times has the delegation or transfer been carried out?

A large part of this attribute can be implemented at the policy level with no loss of generality. Where the constraint description should be lodged depends entirely on the convenience and ease of management.

4 Subsidiarity in Roles

The word *subsidiarity* is defined as "*the principle of devolving political decisions to the lowest practical level.*" in the Collins Concise Dictionary. We use this word to mean that tasks should be done by the appropriate role holder with suitable responsibility-capability, *independent* of the structural authority in the organisation. This is a very important requirement in many organisations. Theoretically, role hierarchy, which is commonly favoured by most researchers in this area, is a wonderful structured construct. It seems that creating a role hierarchy tends to help organise roles according to their shared capabilities (permissions). Unfortunately, subsidiarity has a disruptive influence on this theory.

Often subsidiarity creates counter examples that will show the ungainliness of inheritance orientated construct. While this is not surprising because things don't always fall into a structure, it is surprising how uncommon a useful role hierarchy is in actual organisations, in terms of the concept of inheritance being neatly applicable as suggested in [Sandhu96]. Practical cases we have encountered also have the inconvenient trait of creating a hierarchy in which a role inherits the capabilities of another, only to also have a large number of exclusions due to subsidiarity.

Take the software project example, each role has their own responsibilities that determine their capabilities. The project manager would not expect to have capabilities that allow them to alter designs, code or test specs. They would, instead, have different capabilities relating to project management tasks. It is not a good idea for the project manager to change code; they probably lack the necessary qualifications. A supervising engineer would not even expect to inherit all the capabilities of the engineers they are supervising unless their supervisory role is additional to the standard software engineers role. The supervising engineer would be expected to read code, design, and test results to ensure the correct quality and functionality; they would get the software engineers they supervise to change the code when necessary.

While there are areas in which the collection of capabilities under a single role allows some form of extraction to form a hierarchy, this facility is not the most obvious and convenient approach, as shown in [Bartholdt98]. Another area from which we should be able to draw some real requirements are international standards, such as ISO 9000. A company following these standards should aim to identify the processes along with those roles that are responsible for each stage in the process. Within a project, such as a software project [Mazza96] the project manager would create a project plan including role definitions, the relationships between the roles and the qualifications of the staff to fulfil their role. The software procedures will often determine the capabilities required by a each role. For example, a software engineer (but not the project manager) may have the capability to sign a test acceptance form for other engineers in the team. Only the team leader would have the capability to authorise major design changes.

If role hierarchy is not used, how would RBAC develop? Perhaps there is no impact at all. We believe that further investigation in this area by collecting a large number of practical operational cases will help further understanding in this aspect.

5 Discussion and Summary

A large number of concepts brought out in this paper can, at first glance, be thought of as a policy matter. For example, the condition for the activation of a role may be regarded as one that pertains to the policy governing the role. However, as we have seen, the role of fire marshal, for example, is dependent on the event of a fire, or the sounding of the fire alarm. To relegate such intrinsic attributes to a policy outside the definition of role does not seem natural. In cases where the role and policy administrators are different it is essential to keep role-related attributes in the domain of the role administrator.

One major limitation of existing RBAC approaches is to treat role parochially as a database centric problem. If

we consider web-based electronic commerce over the internet, database access is only one of the many access control problems. Complexity in the environment calls for better management, which can be helped by having a more complete role model and the judicious use of some part of the full model. We further this consideration by proposing the use of logical constraints that specify a relevant state that is important to a role, rather than simple primitives such as strings and integers. This is a much more powerful and flexible way to fully describe roles. This is shown in a number of items in section 3 and we believe that incorporation of this approach can also enable a closer integration with RBAC management as well as the enforcement of other system level policy. Clearly, the response time for making an access decision could be critical in some cases. This implies that only a minimal set of the full role model should be adopted in accordance to good engineering practices.

When the necessary constraints of a role are taken into account, it becomes clear that role combination as found in the real world does not follow a nice hierarchical structure as found in theory. Organisations appoint people to a role which may appear at first as one that inherits from different roles. But, when roles with different activation constraints, for example, are aggregated, it will be difficult to derive a simple activation constraint for that aggregation. It means that creating a hierarchy for roles that are fully specified is likely to create great complexity that may not be justifiable or desirable from the management's point of view.

From these cases, we come to the conclusion that the concept of hierarchy should be used with caution in the discussion of roles, given that very often this translates badly into practical application, as has been outlined in section 4. It is hoped that with increased understanding, the judicious application, or abstinence of its application, will enable better use and management of RBAC.

References

[Bartholdt98] Bartholdt, J., *TrustCenter–Secure Web with Role-based Access Control Mechanisms and Secure Email Environment*, Proceedings of the 5th HP-OVUA Workshop, '98, ENST-Bretagne, Rennes, France, 19-21 April 1998.

[Bertiino97] Bertino, E., and Ferrari, E., *Administration Policies in a Multipolicy Authorization System*, Proceedings of the XIth annual working conference on Database security, IFIP '97.

[Chen96] Chen, F. and Sandhu, R. S., *Constraints for Role-Based Access Control*, Proceedings of the 1st ACM RBAC Workshop, Githersburg, MD, USA, 30 November–1 December 1995.

[DMTF-CIM] Desktop Management Task Force, *Common Information Model (CIM) Version 2.0 (Draft)*, March, 1998

[Giuri97] Giuri, L. and Iglio, P., *Role templates for content based access control* Proceedings of the 2nd ACM RBAC Workshop, Fairfax, VA, USA, 6-7 November 1997

[Goh97] Goh, C., *A Generic Approach to Policy Description in System Management*, Proceedings of the 8th IFIP/IEEE international workshop on Distributed Systems Operations & Management (DSOM '97), Sydney, Australia, 21-23 October 1997.

[Lawrence93] Lawrence, L. G., *The Role of Roles*, Computers & Security, Vol. 12 No 1, 1993.

[Lupu97] Lupu, E. and Sloman, M., *Reconciling Role Based Management and Role Based Access Control*, Proceedings of the 2nd ACM RBAC Workshop, Fairfax, VA, USA, 6-7 November 1997

[Lupu98] Lupu, E. and Sloman, M., *A Policy Based Role Object Model*, Proceedings of the 5th HP-OVUA Workshop, '98, ENST-Bretagne, Rennes, France, 19-21 April 1998.

[Mazza96] Mazza, C., Fairclough, J., Melton, B., de Pablo, D., Scheffer A., Stevens, R., Jones, M., Alvisi, G. Software engineering Guides. Prentice Hall 1996

[Moffett94] Moffett, J.D., *Network and Distributed Systems Management*, editor Sloman, Addison-Wesley, 1994, chapter 17.

[Sandhu96] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman C. E., *Role-Based Access Control Models*, IEEE Computer, Vol 29, No. 2, Feb 1996, pp 38–47.

[Sloman94] Sloman, M., *Policy Driven Management for Distributed Systems*, Journal of Network and Systems Management, vol. 2 part 4, 1994, pp333-60.

[Yialelis96] Yialelis, N., Sloman, M., *A Security Framework Supporting Domain Based Access Control in Distributed Systems*, ISOC Symposium on Network and Distributed Systems Security (SNDSS96), San Diego, CA, USA, February 1996.