A Spatio-Temporal Role-Based Access Control Model

Indrakshi Ray and Manachai Toahchoodee

Department of Computer Science
Colorado State University
{iray,toahchoo}@cs.colostate.edu

Abstract. With the growing advancement of pervasive computing technologies, we are moving towards an era where spatio-temporal information will be necessary for access control. The use of such information can be used for enhancing the security of an application, and it can also be exploited to launch attacks. For critical applications, a formal model for spatio-temporal-based access control is needed that increases the security of the application and ensures that the location information cannot be exploited to cause harm. In this paper, we propose a spatio-temporal access control model, based on the Role-Based Access Control (RBAC) model, that is suitable for pervasive computing applications. We show the association of each component of RBAC with spatio-temporal information. We formalize the model by enumerating the constraints. This model can be used for applications where spatial and temporal information of a subject and an object must be taken into account before granting or denying access.

1 Introduction

With the increase in the growth of wireless networks and sensor and mobile devices, we are moving towards an era of pervasive computing. The growth of this technology will spawn applications such as, the Aware Home [5] and CMU's Aura [7], that will make life easier for people. Pervasive computing applications introduce new security issues that cannot be addressed by existing access control models and mechanisms. For instance, access to a computer should be automatically disabled when a user walks out of the room. Traditional models, such as Discretionary Access Control (DAC) or Role-Based Access Control (RBAC) do not take into account such environmental factors in determining whether access should be allowed or not. Consequently, new access control models and mechanisms are needed that use environmental factors, such as, time and location, while determining access.

The use of spatial and temporal information for access can be used for enhancing the security of other applications as well. For instance, a user should be able to fire a missile from specific high security locations only. Moreover, the missile can be fired only when it is in a certain location. For such critical applications, we can include additional checks, such as the verification of the location of the user and that of the missile, that must be satisfied before the user is granted access. With the reduction in the cost of Global Positioning System, this is indeed a viable option.

In this paper, we propose a formal spatio-temporal model that is suitable for commercial applications. Since RBAC is policy-neutral, simplifies access management, and

used by commercial applications, we decide to base our work on it. We show how RBAC can be extended to incorporate the notion of time and location. We illustrate how each component of RBAC is related with time and location. In other words, we explain how time and location impact each component of RBAC. Finally, we show how this spatio-temporal information can be used to determine whether an user has access to a given object. The correct behavior of this model is formulated in terms of constraints that must be satisfied by any application using this model.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 briefly illustrates how time and location are represented in our model. Section 4 shows the relationship of each component of Core RBAC with location. Sections 5, 6 and 7 propose different types of hierarchies and separation of duty constraints that we can have in our model. Section 8 discusses a simple example using our model. Section 9 concludes the paper with some pointers to future directions.

2 Related Work

Recently, some research attempts have been done under the area of context aware access control. Yu et al. [15] proposed LTAM a location-temporal authorization model, which is based on a Discretionary Access Control (DAC) model. The goal of this paper is different than ours; it focuses on controlling access to the different locations. For example, access rules may have temporal constraints that can specify when a user can enter or leave a location or how many times a user can enter a location. However, it does not address the issue of where and when a subject can access a given object. Since this model is based on DAC, authorization management is a problem.

Role-based access control model [6] is used for addressing the access control needs of commercial organizations. In RBAC permissions are attached to roles and users must be assigned to roles to get the permissions. Permissions determine what operations can be carried out on resources under access control. A user must establish a session to activate a subset of roles to which the user is assigned. Each user can activate multiple sessions, however, each session is associated with only one user. The operations that a user can perform in a session depend on the roles activated in that session and the permissions associated with those roles. RBAC also supports role hierarchies. Role hierarchies define an inheritance relationship between roles. To prevent conflict of interests that arise in an organization, RBAC allows the specification of Static and Dynamic Separation of Duty constraints.

Several work exists that improve RBAC functionality. We discuss only those that are very closely related to ours. Some of these work focus on how RBAC can be extended to make it context aware. Sampemane et al. [12] present a new access control model for active spaces. Active space denotes the computing environment integrating physical spaces and embedded computing software and hardware entities. The active space allows interactive exchange of information between the user and the space. Environmental aspects are adopted into the access control model for active spaces, and the space roles are introduced into the implementation of the access control model based on RBAC. The model supports specification of MAC policies in which system admin-

istrator maintains the access matrix and DAC policies in which users create and update security policies for their devices.

Covington et al. [5] introduce environment roles in a generalized RBAC model (GRBAC) to help control access control to private information and resources in ubiquitous computing applications. The environments roles differ from the subject roles in RBAC but do have similar properties including role activation, role hierarchy and separation of duty. In the access control framework enabled by environment roles, each element of permission assignment is associated with a set of environment roles, and environment roles are activated according to the changing conditions specified in environmental conditions; in this way, environmental properties like time and location are introduced to the access control framework. In a subsequent work [4], Covington et al. describes the Context-Aware Security Architecture (CASA) which is an implementation of the GRBAC model. The access control is provided by the security services in the architecture. In CASA, polices are expressed as roles and managed by the security management service, authentication and authorization services are used to verify user credentials and determine access to the system resources. The environmental role activation services manage environmental role activation and deactivation according to the environment variables collected by the context management services.

Other extensions to RBAC include the Temporal Role-Based Access Control Model (TRBAC) proposed by Bertino et al. [1]. This work adds the time dimension to the RBAC model. The authors in this paper introduce the concept of role enabling and disabling. Temporal constraints determine when the roles can be enabled or disabled. A role can be activated only if it has been enabled. Joshi et al.[8] extend this work by proposing the Generalized Temporal Role Based Access Control Model (GTRBAC). In this work the authors introduce the concept of time-based role hierarchy and time-based separation of duty. These works do not discuss the impact of spatial information.

Researchers have also extended RBAC to incorporate spatial information. The most important work in this regard is the GEO-RBAC [2]. In this model, role activation is based on the location of the user. For instance, a user can acquire the role of teacher only when he is in the school. Outside the school, he can acquire the role of citizen. The model supports role hierarchies but does not deal with separation of duties. Another work incorporating spatial information is by Ray et al. [11]. Here again, the authors propose how each component of RBAC is influenced by location. The authors define their formal model using the Z specification language. Role hierarchy and separation of duties are not addressed in this paper. None of these work discuss the impact of time on location. Location-based access control has been addressed in other works not pertaining to RBAC [7, 9, 10].

To the best of our knowledge, the only work which propose incorporating both time and location in RBAC is by Chandran et al. [3]. The paper combines the main features of GTRBAC and GEO-RBAC. Here again, role is enabled by time constraints. The user can activate the role if the role is enabled and the user satisfies the location constraints associated with role activation. Our current work is closely related to this work. The similarity is that in both the models role activation occurs when temporal and spatial constraints are satisfied. However, there are a number of points where we differ. First, in Chandran's work, role assignment is not dependent on location or time. A number

of motivating examples indicate that role assignment should be dependent on role and time. Consequently, we incorporate this feature in our model. Second, in Chandran's work, when a role can be activated all the permissions associated with the role can be invoked. This may not be true in real world. For instance, a system administrator's role can be activated from 9:00 a.m. to 9:00 p.m. everyday. However, he can perform backup only during 8:00 to 9:00 p.m. on Fridays. Chandran's model cannot express this situation. We associate a permission with additional location and temporal constraints that must be satisfied before a permission can be invoked. Third, Chandran's work does not discuss the impact of location and time on role hierarchy or separation of duty. We propose different types of time and location based hierarchy and separation of duty constraints in our model which will be useful for real-world applications.

3 Representing Location and Time

3.1 Representing Location

In order to perform location-based access control, we need to perform operations on location information and protect the location information. In this section, we formalize the concept of location [2, 3] and propose the location comparison operators that are used in our model.

There are two types of locations: *physical* and *logical*. All users and objects are associated with locations that correspond to the physical world. These are referred to as the physical locations. A physical location is formally defined by a set of points in a three-dimensional geometric space.

Definition 1. [Physical Location] *A physical location $PLoc_i$ is a non-empty set of points $\{p_i, p_j, \ldots, p_n\}$ where a point p_k is represented by three co-ordinates.*

Physical locations are grouped into symbolic representations that will be used by applications. We refer to these symbolic representations as logical locations. Examples of logical locations are Fort Collins, Colorado etc.

Definition 2. [Logical Location] *A logical location is an abstract notion for one or more physical locations.*

We assume the existence of two translation functions, m and m', that convert from logical locations to physical locations and vice-versa.

Definition 3. [Mapping Functions m and m'] *m is a total function that converts a physical location into a logical one. m' is a total function that converts a logical location into a physical one. Let P be the set of all possible physical locations and L be the set of all logical locations. The following formalizes the functions.*

- $m : P \rightarrow L$.
- $m' : L \rightarrow P$
- *For any logical location Loc_i, $m(m'(Loc_i)) = Loc_i$.*
- *For any physical location $PLoc_j$, $m'(m(PLoc_j)) = PLoc_j$.*

Different kinds of relationships may exist between a pair of locations. We discuss one such relationship, known as *containment*, that will be used in this paper. Containment formalizes the idea whether one location is contained within another. Intuitively, a physical location $ploc_j$ is contained in another physical location $ploc_k$, if all points in $ploc_j$ also belong to $ploc_k$. This is formalized as follows.

Definition 4. [Containment Relation] *A physical location $ploc_j$ is said to be contained in another physical location $ploc_k$, denoted as, $ploc_j \subseteq ploc_k$, if the following condition holds: $\forall p_i \in ploc_j, p_i \in ploc_k$. The location $ploc_j$ is called the contained location and $ploc_k$ is referred to as the containing or the enclosing location. A logical location $lloc_m$ is contained in $lloc_n$, denoted as, $lloc_m \subseteq lloc_n$, if and only if the physical location corresponding to $lloc_m$ is contained within that of $lloc_m$, that is $m'(lloc_m) \subseteq m'(lloc_n)$.*

Note that, a physical location may be contained in a logical location or vice-versa. In such cases, we use the mapping functions to convert the logical locations into physical ones and then test whether one is contained within the other.

We assume the existence of a logical location called *universe* that contains all other locations.

In the rest of the paper, we do not discuss physical locations any more. The locations referred to are logical locations.

3.2 Representing Time

Our model uses two kinds of temporal information. We feel it is necessary to distinguish between these two kinds of information because they have very different semantics. The first is known as time instant and the other is time interval. Time can be represented as a set of discrete points on the time line.

Definition 5. [Time Instant] *A* time instant *is one discrete point on the time line.*

The exact granularity of a time instant will be application dependent. For instance, in some application a time instant may be measured at the nanosecond level and in another one it may be specified at the millisecond level.

Definition 6. [Time Interval] *A* time interval *is a set of time instances. When the time instances making up an interval are consecutive, we refer to the interval as a* continuous *one. Otherwise, the interval is said to be* non-continuous.

Example of a continuous interval is 9:00 a.m. to 3:00 p.m. on 25th December. Example of a non-continuous interval is 9:00 a.m. to 6:00 p.m. on Mondays to Fridays in the month of March. Some researchers refer to time intervals as time expressions. We use the notation $t_i \in d$ to mean that t_i is a time instance in the time interval d.

Two time intervals can be related by any of the following relations: *disjoint*, *equality*, and *overlapping*. Two time intervals tv_i and tv_j are disjoint if the intersection of the set of time instances in tv_i with those of tv_j results in the null set. Two time intervals tv_i and tv_j are equal if the set of time instances in tv_i is equal to those of tv_j. Two time intervals tv_i and tv_j are overlapping if the intersection of the set of time instances in

tv_i with those of tv_j results in a non-empty set. A special case of overlapping relation is referred to as *containment*. A time interval tv_i is contained in another interval tv_j if the set of time instances in tv_i is a subset of those in tv_j. We formally denote this as $tv_i \preceq tv_j$.

4 Relationship of Core-RBAC Entities with Time and Location

In this section, we describe how the entities in RBAC are associated with location and time. The different entities of RBAC are *Users*, *Roles*, *Sessions*, *Permissions*, *Objects* and *Operations*. We discuss how each of these components are associated with location and time.

4.1 Users

We assume that each valid user, interested in doing some location-sensitive operation, carries a locating device which is able to track his location. The location of a user changes with time. The relation $UserLocation(u,t)$ gives the location of the user at any given time instant t. Since a user can be associated with only one location at any given point of time, we have the following constraint:

$$UserLocation(u,t) = l_i \wedge UserLocation(u,t) = l_j \Leftrightarrow (l_i \subseteq l_j) \vee (l_j \subseteq l_i)$$

We define a similar function $UserLocations(u,d)$ that gives the location of the user during the time interval d. Note that, a single location can be associated with multiple users at any given point of time.

4.2 Objects

Objects can be physical or logical. Example of a physical object is a computer. Files are examples of logical objects. Physical objects have devices that transmit their location information with the timestamp. Logical objects are stored in physical objects. The location and timestamp of a logical object corresponds to the location and time of the physical object containing the logical object. We assume that each object is associated with one location at any given instant of time. Each location can be associated with many objects. The function *ObjLocation(o,t)* takes as input an object o and a time instance t and returns the location associated with the object at time t. Similarly, the function *ObjLocations(o,d)* takes as input an object o and time interval d and returns the location associated with the object.

4.3 Roles

We have three types of relations with roles. These are user-role assignment, user-role activation, and permission-role assignment.

We begin by focusing on user-role assignment. Often times, the assignment of user to roles is location and time dependent. For instance, a person can be assigned the role of U.S. citizen only in certain designated locations and at certain times only. To get the

role of conference attendee, a person must register at the conference location during specific time intervals. Thus, for a user to be assigned a role, he must be in designated locations during specific time intervals. In our model, a user must satisfy spatial and temporal constraints before roles can be assigned. We capture this with the concept of *role allocation*. A role is said to be *allocated* when it satisfies the temporal and spatial constraints needed for role assignment. A role can be assigned once it has been allocated. $RoleAllocLoc(r)$ gives the set of locations where the role can be allocated. $RoleAllocDur(r)$ gives the time interval where the role can be allocated. Some role s can be allocated anywhere, in such cases $RoleAllocLoc(s) = universe$. Similarly, if role p can be assigned at any time, we specify $RoleAllocDur(p) = always$.

Some roles can be activated only if the user is in some specific locations. For instance, the role of audience of a theater can be activated only if the user is in the theater when the show is on. The role of conference attendee can be activated only if the user is in the conference site while the conference is in session. In short, the user must satisfy temporal and location constraints before a role can be activated. We borrow the concept of *role-enabling* [1, 8] to describe this. A role is said to be *enabled* if it satisfies the temporal and location constraints needed to activate it. A role can be activated only if it has been enabled. $RoleEnableLoc(r)$ gives the location where role r can be activated and $RoleEnableDur(r)$ gives the time interval when the role can be activated.

The predicate $UserRoleAssign(u,r,d,l)$ states that the user u is assigned to role r during the time interval d and location l. For this predicate to hold, the location of the user when the role was assigned must be in one of the locations where the role allocation can take place. Moreover, the time of role assignment must be in the interval when role allocation can take place.

$$UserRoleAssign(u,r,d,l) \Rightarrow (UserLocation(u,d) = l) \wedge (l \subseteq RoleAllocLoc(r)) \wedge (d \subseteq RoleAllocDur(r))$$

The predicate $UserRoleActivate(u,r,d,l)$ is true if the user u activated role r for the interval d at location l. This predicate implies that the location of the user during the role activation must be a subset of the allowable locations for the activated role and all times instances when the role remains activated must belong to the duration when the role can be activated and the role can be activated only if it is assigned.

$$UserRoleActivate(u,r,d,l) \Rightarrow (l \subseteq RoleEnableLoc(r)) \wedge (d \subseteq RoleEnableDur(r)) \wedge UserRoleAssign(u,r,d,l)$$

The additional constraints imposed upon the model necessitates changing the preconditions of the functions *AssignRole* and *ActivateRole*.

The permission role assignment is discussed later.

4.4 Sessions

In mobile computing or pervasive computing environments, we have different types of sessions that can be initiated by the user. Some of these sessions can be location-dependent, others not. Thus, sessions are classified into different types. Each instance of a session is associated with some type of a session. The type of session instance s is

given by the function $Type(s)$. The type of the session determines the allowable location. The allowable location for a session type st is given by the function $SessionLoc(st)$.

When a user u wants to create a session si, the location of the user for the entire duration of the session must be contained within the location associated with the session. The predicate $SessionUser(u,s,d)$ indicates that a user u has initiated a session s for duration d.

$$SessionUser(u,s,d) \Rightarrow (UserLocation(u,d) \subseteq SessionLoc(Type(s)))$$

Since sessions are associated with locations, not all roles can be activated within some session. The predicate $SessionRoles(u,r,s,d,l)$ states that user u initiates a session s and activates a role for duration d and at location l.

$$SessionRole(u,r,s,d) \Rightarrow UserRoleActivate(u,r,d,l) \wedge l \subseteq SessionLoc(Type(s)))$$

4.5 Permissions

The goal of our model is to provide more security than their traditional counterparts. This happens because the time and location of a user and an object are taken into account before making the access decisions. Our model also allows us to model real-world requirements where access decision is contingent upon the time and location associated with the user and the object. For example, a teller may access the bank confidential file if and only if he is in the bank and the file location is the bank secure room and the access is granted only during the working hours. Our model should be capable of expressing such requirements.

Permissions are associated with roles, objects, and operations. We associate three additional entities with permission to deal with spatial and temporal constraints: user location, object location, and time. We define three functions to retrieve the values of these entities. $PermRoleLoc(p,r)$ specifies the allowable locations that a user playing the role r must be in for him to get permission p. $PermObjLoc(p,o)$ specifies the allowable locations that the object o must be in so that the user has permission to operate on the object o. $PermDur(p)$ specifies the allowable time when the permission can be invoked.

We define another predicate which we term $PermRoleAcquire(p,r,d,l)$. This predicate is true if role r has permission p for duration d at location l. Note that, for this predicate to be true, the time interval d must be contained in the duration where the permission can be invoked and the role can be enabled. Similarly, the location l must be contained in the places where the permission can be invoked and role can be enabled.

$$PermRoleAcquire(p,r,d,l) \Rightarrow (l \subseteq (PermRoleLoc(p,r) \cap RoleEnableLoc(r))) \\ \wedge (d \subseteq (PermDur(p) \cap RoleEnableDur(p)))$$

The predicate $PermUserAcquire(u,o,p,d,l)$ means that user u can acquire the permission p on object o for duration d at location l. This is possible only when the permission p is assigned some role r which can be activated during d and at location l, the user location and object location match those specified in the permission, the duration d matches that specified in the permission.

$$PermRoleAcquire(p,r,d,l) \wedge UserRoleActivate(u,r,d,l) \\ \wedge (ObjectLocation(o,d) \subseteq PermObjectLoc(p,o)) \Rightarrow PermUserAcquire(u,o,p,d,l)$$

5 Impact of Time and Location on Role-Hierarchy

The structure of an organization in terms of lines of authority can be modeled as an hierarchy. This organization structure is reflected in RBAC in the form of a role hierarchy [13]. Role hierarchy is a relation among roles. This relation is transitive, and anti-symmetric. Roles higher up in the hierarchy are referred to as senior roles and those lower down are junior roles. The major motivation for adding role hierarchy to RBAC was to simplify role management. Senior roles can inherit the permissions of junior roles, or a senior role can activate a junior role, or do both depending on the nature of the hierarchy. This obviates the need for separately assigning the same permissions to all members belonging to a hierarchy.

Joshi et al. [8] identify two basic types of hierarchy. The first is the permission inheritance hierarchy where a senior role x inherits the permission of a junior role y. The second is the role activation hierarchy where a user assigned to a senior role can activate a junior role. Each of these hierarchies may be constrained by location and temporal constraints. Consequently, we have a number of different hierarchical relationships in our model.

Definition 7. [Unrestricted Permission Inheritance Hierarchy] *Let x and y be roles such that $x \geq y$, that is, senior role x has an unrestricted permission-inheritance relation over junior role y. In such a case, x inherit's y's permissions but not the locations and time associated with it. This is formalized as follows:*

$$\forall p, (x \geq y) \wedge PermRoleAcquire(p, y, d, l) \Rightarrow PermRoleAcquire(p, x, d', l')$$

In the above hierarchy, a senior role inherits the junior roles permissions. However, unlike the junior role, these permissions are not restricted to time and location. Account auditor role inherits the permissions from the accountant role. He can use the permissions at any time and at any place.

Definition 8. [Unrestricted Activation Hierarchy] *Let x and y be roles such that $x \succcurlyeq y$, that is, senior role x has a role-activation relation over junior role y. Then, a user assigned to role x can activate role y at any time and at any place. This is formalized as follows:*

$$\forall u, (x \succcurlyeq y) \wedge UserRoleActivate(u, x, d, l) \Rightarrow UserRoleActivate(u, y, d', l')$$

Here again a user who can activate a senior role can also activate a junior role. This junior role can be activated at any time and place. A project manager can activate the code developer role at any time and at any place.

Definition 9. [Time Restricted Permission Inheritance Hierarchy] *Let x and y be roles such that $x \geq_t y$, that is, senior role x has a time restricted permission-inheritance relation over junior role y. In such a case, x inherit's y's permissions together with the temporal constraints associated with the permission. This is formalized as follows:*

$$\forall p, (x \geq_{lt} y) \wedge PermRoleAcquire(p, y, d, l) \Rightarrow PermRoleAcquire(p, x, d, l')$$

In the above hierarchy, a senior role inherits the junior roles permissions. However, the duration when the permissions are valid are those that are associated with the junior roles. A contact author can inherit the permissions of the author until the paper is submitted.

Definition 10. [Time Restricted Activation Hierarchy] *Let x and y be roles such that $x \succcurlyeq_t y$, that is, senior role x has a role-activation relation over junior role y. Then, a user assigned to role x can activate role y only at the time when role y can be enabled. This is formalized as follows:*

$$\forall u,\ (x \succcurlyeq_{tl} y) \wedge UserRoleActivate(u,x,d,l) \wedge d \subseteq RoleEnableDur(y) \Rightarrow UserRoleActivate(u,y,d,l')$$

Here again a user who can activate a senior role can also activate a junior role. However, this activation is limited to the time when the junior role can be activated. A program chair can activate a reviewer role only during the review period.

Definition 11. [Location Restricted Permission Inheritance Hierarchy] *Let x and y be roles such that $x \geq_l y$, that is, senior role x has a location restricted permission-inheritance relation over junior role y. In such a case, x inherit's y's permissions together with the location constraints associated with the permission. This is formalized as follows:*

$$\forall p, (x \geq_{lt} y) \wedge PermRoleAcquire(p,y,d,l) \Rightarrow PermRoleAcquire(p,x,d',l)$$

In the above hierarchy, a senior role inherits the junior roles permissions. These permissions are restricted to the locations imposed on the junior roles. A top secret scientist inherits the permission of top secret citizen only when he is in top secret locations.

Definition 12. [Location Restricted Activation Hierarchy] *Let x and y be roles such that $x \succcurlyeq_l y$, that is, senior role x has a role-activation relation over junior role y. Then, a user assigned to role x can activate role y only at the places when role y can be enabled. This is formalized as follows:*

$$\forall u,\ (x \succcurlyeq_{tl} y) \wedge UserRoleActivate(u,x,d,l) \wedge l \subseteq RoleEnableLoc(y) \Rightarrow UserRoleActivate(u,y,d',l)$$

Here again a user who can activate a senior role can also activate a junior role. However, this activation is limited to the place where the junior role can be activated. A Department Chair can activate a Staff role only when he is in the Department.

Definition 13. [Time Location Restricted Permission Inheritance Hierarchy] *Let x and y be roles such that $x \geq y$, that is, senior role x has a time-location restricted permission-inheritance relation over junior role y. In such a case, x inherit's y's permissions together with the temporal and location constraints associated with the permission. This is formalized as follows:*

$$\forall p, (x \geq_{lt} y) \wedge PermRoleAcquire(p,y,d,l) \Rightarrow PermRoleAcquire(p,x,d,l)$$

In the above hierarchy, a senior role inherits the junior roles permissions. These permissions are restricted to time and locations imposed on the junior roles. Daytime doctor role inherits permission of daytime nurse role only when he is in the hospital during the daytime.

Definition 14. [Time Location Restricted Activation Hierarchy] *Let x and y be roles such that $x \succcurlyeq_{tl} y$, that is, senior role x has a role-activation relation over junior role y. Then, a user assigned to role x can activate role y only at the places and during the time when role y can be enabled. This is formalized as follows:*

$$\forall u,\ (x \succcurlyeq_{tl} y) \wedge UserRoleActivate(u,x,d,l) \wedge d \subseteq RoleEnableDur(y) \\ \wedge l \subseteq RoleEnableLoc(y) \Rightarrow UserRoleActivate(u,y,d,l)$$

Here again a user who can activate a senior role can also activate a junior role. However, this activation is limited to the time and place where the junior role can be activated. User who has a role of mobile user can activate the weekend mobile user role only if he/she is in the US during the weekend.

It is also possible for a senior role and a junior role to be related with both permission inheritance and activation hierarchies. In such a case, the application will choose the type of inheritance hierarchy and activation hierarchy needed.

6 Impact of Time and Location on Static Separation Of Duties

Separation of duties (SoD) enables the protection of the fraud that might be caused by the user [14]. SoD can be either static or dynamic. Static Separation of Duty (SSoD) comes in two varieties. First one is with respect to user role assignment. The second one is with respect to permission role assignment. In this case, the SoD is specified as a relation between roles. The idea is that the same user cannot be assigned to the same role. Due to the presence of temporal and spatial constraints, we can have different flavors of separation of duties – some that are constrained by temporal and spatial constraints and others that are not. In the following we describe the different separation of duty constraints.

Definition 15. [Weak Form of SSoD - User Role Assignment] *Let x and y be two roles such that $x \neq y$. $x,y \in SSOD_w(ROLES)$ if the following condition holds:*

$$UserRoleAssign(u,x,d,l) \Rightarrow \neg\ UserRoleAssign(u,y,d,l)$$

The above definition says that a user u assigned to role x during time d and location l cannot be assigned to role y at the same time and location if x and y are related by $SSOD_w$. An example where this form is useful is that a user should not be assigned the audience role and mobile user role at the same time and location.

Definition 16. [Strong Temporal Form of SSoD - User Role Assignment] *Let x and y be two roles such that $x \neq y$. $(x,y) \in SSOD_t(ROLES)$ if the following condition holds:*

$$UserRoleAssign(u,x,d,l) \Rightarrow \neg\ (\exists d' \subseteq always \bullet UserRoleAssign(u,y,d',l))$$

The above definition says that a user u assigned to role x during time d and location l cannot be assigned to role y at any time in the same location if x and y are related by $SSOD_t$. The consultant for oil company A will never be assigned the role of consultant for oil company B in the same country.

Definition 17. [Strong Spatial Form of SSoD - User Role Assignment] *Let x and y be two roles such that $x \neq y$. $(x,y) \in SSOD_l(ROLES)$ if the following condition holds:*

$$UserRoleAssign(u,x,d,l) \Rightarrow \neg\ (\exists l' \subseteq universe \bullet UserRoleAssign(u,y,d,l'))$$

The above definition says that a user u assigned to role x during time d and location l, he cannot be assigned to role y at the same time at any location if x and y are related by $SSOD_l$. A person cannot be assigned the roles of realtor and instructor at the same time.

Definition 18. [Strong Form of SSoD - User Role Assignment] *Let x and y be two roles such that $x \neq y$. $(x,y) \in SSOD_s(ROLES)$ if the following condition holds:*

$$UserRoleAssign(u,x,d,l) \Rightarrow \neg\ (\exists l' \subset universe, \exists d' \subseteq always \bullet UserRoleAssign(u,y,d',l'))$$

The above definition says that a user u assigned to role x during time d and location l, he cannot be assigned to role y at any time or at any location if x and y are related by $SSOD_s$. The same employee cannot be assigned the roles of male and female employee at any given corporation.

We next consider the second form of static separation of duty that deals with permission role assignment. The idea is that the same role should not acquire conflicting permissions. The same manager should not make a request for funding as well as approve it.

Definition 19. [Weak Form of SSoD - Permission Role Assignment] *Let p and q be two permissions such that $p \neq q$. $(p,q) \in SSOD\_PRA_w$ if the following condition holds:*

$$PermRoleAcquire(p,x,d,l) \Rightarrow \neg\ PermRoleAcquire(q,x,d,l))$$

The above definition says that if permissions p and q are related through weak SSoD Permission Role Assignment and x has permission p at time d and location l, then x should not be given permission q at the same time and location.

Definition 20. [Strong Temporal Form of SSoD - Permission Role Assignment] *Let p and q be two permissions such that $p \neq q$. $(p,q) \in SSOD\_PRA_t$ if the following condition holds:*

$$PermRoleAcquire(p,x,d,l) \Rightarrow \neg\ (\exists d' \subseteq always \bullet PermRoleAcquire(q,x,d',l))$$

The above definition says that if permissions p and q are related through strong temporal SSoD Permission Role Assignment and x has permission p at time d and location l, then x should not get permission q at any time in location l.

Definition 21. [Strong Spatial Form of SSoD - Permission Role Assignment] *Let p and q be two permissions such that $p \neq q$. $(p,q) \in SSOD\_PRA_t$ if the following condition holds:*

$PermRoleAcquire(p,x,d,l) \Rightarrow \neg\ (\exists l' \subset universe \bullet PermRoleAcquire(q,x,d,l'))$

The above definition says that if permissions p and q are related through strong spatial SSoD Permission Role Assignment and x has permission p at time d and location l, then x should not be given permission q at the same time.

Definition 22. [Strong Form of SSoD - Permission Role Assignment] *Let p and q be two permissions such that $p \neq q$. $(p,q) \in SSOD\_PRA_s$ if the following condition holds:*

$PermRoleAcquire(p,x,d,l) \Rightarrow \neg\ (\exists l' \subset universe, \exists d' \subseteq always \bullet PermRoleAcquire(q,x,d',l'))$

The above definition says that if permissions p and q are related through strong SSoD Permission Role Assignment, then the same role should never be given the two conflicting permissions.

7 Impact of Time and Location on Dynamic Separation of Duties

Static separation of duty ensures that a user does not get assigned conflicting roles or a role is not assigned conflicting permissions. Dynamic separation of duty addresses the problem that a user is not able to activate conflicting roles during the same session.

Definition 23. [Weak Form of DSoD] *Let x and y be two roles such that $x \neq y$. $(x,y) \in DSOD_s$ if the following condition holds:*

$SessionRole(u,x,s,d,l) \Rightarrow \neg\ SessionRole(u,y,s,d,l))$

The above definition says that if roles x and y are related through weak DSoD and if user u has activated role x in some session s for duration d and location l, then u cannot activate role y during the same time and in the same location in session s. In the same session, a user can activate a sales assistant role and a customer role. However, both these roles should not be activated at the same time in the same location.

Definition 24. [Strong Temporal Form of DSoD] *Let x and y be two roles such that $x \neq y$. $(x,y) \in DSOD_s$ if the following condition holds:*

$SessionRole(u,x,s,d,l) \Rightarrow \neg\ (\exists d' \subset always, \bullet SessionRole(u,y,s,d',l))$

The above definition says that if roles x and y are related through strong temporal DSoD and if user u has activated role x in some session s, then u can never activate role y any time at the same location in the same session. In a teaching session in a classroom, a user cannot activate the the grader role once he has activated the student role.

Definition 25. [Strong Spatial Form of DSoD] *Let x and y be two roles such that $x \neq y$. $(x,y) \in DSOD_l$ if the following condition holds:*

$SessionRole(u,x,s,d,l) \Rightarrow \neg\ (\exists l' \subseteq universe \bullet SessionRole(u,y,s,d,l'))$

The above definition says that if roles x and y are related through strong DSoD and if user u has activated role x in some session s, then u can never activate role y in session s during the same time in any location. If a user has activated the Graduate Teaching Assistant role in his office, he cannot activate the Lab Operator role at the same time.

Definition 26. [Strong Form of DSoD] *Let x and y be two roles such that $x \neq y$. $(x,y) \in DSOD_s$ if the following condition holds:*

$$SessionRole(u,x,s,d,l) \Rightarrow \neg\ (\exists l' \subset universe, \exists d' \subseteq always \bullet SessionRole(u,y,s,d',l'))$$

The above definition says that if roles x and y are related through strong DSoD and if user u has activated role x in some session s, then u can never activate role y in the same session. A user cannot be both an code developer and a code tester in the same session.

8 Example Scenario

Example 1. Consider the following access control policy of SECURE bank

1. The organization has five users: Tom, Leena, Diana, Nina and Sam.
2. The organization consists of six roles: teller, loan officer, daytime system operator, nighttime system operator, system operator manager, auditor
3. The teller can read and write teller files only from the teller booth during working hours, that is, 9:00AM - 6:00PM, Monday to Friday.
4. The loan officer can read and write loan files only from the loan office during working hours, that is, 9:00AM - 6:00PM, Monday to Friday.
5. The daytime system operator (DTSO) can backup any file from anywhere in the SECURE bank building during working hours, that is, 9:00AM - 6:00PM, Monday to Friday.
6. The nighttime system operator (NTSO) can backup and restore any file from anywhere in the SECURE bank building during nighttime shift, that is, 6:00PM - 9:00AM, Monday to Friday.
7. The system operator manager (SOM) rights consist of all rights from daytime system operator and night time system operator.
8. The auditor can audit teller files during the working hours.
9. The same person cannot be the teller and the auditor in the same session.
10. Teller files and loan files can be written during working hours only

We can represent the above access control policy using STRBAC.

1. $WorkingHours = \{\{2,3,4,5,6\}.Days + 10.Hours \triangleright 9.Hours\}$
2. $NightTime = \{\{2,3,4,5,6\}.Days + 19.Hours \triangleright 14.Hours\}$
3. Set of Time Intervals $= \{WorkingHours, NighTime\}$
4. Set of locations $= \{TellerBooth, LoanOffice, ComputerRoom, Building\}$
5. $Users = \{Tom, Leena, Diana, Nina, Sam\}$
6. $Roles = \{Teller, LoanOfficer, Auditor, DTSO, NTSO, SOM\}$
7. $RoleEnable = \{$(*Teller, WorkingHours, TellerBooth*), (*Loaner, WorkingHours, LoanerOffice*), (*DTSO, WorkingHours, Building*), (*NTSO, NightTime, Building*), (*SOM, AnyTime, Building*)$\}$
8. *Permissions* consists of
   - $readTellerFile =$ (*Read, TellerFile, AnyTime, TellerBooth, ComputerRoom*)
   - $writeTellerFile =$ (*Write, TellerFile, WorkingHours, TellerBooth, ComputerRoom*)

- $readLoanerFile =$ (*Read, LoanerFile, AnyTime, LoanerOffice, ComputerRoom*)
- $writeLoanerFile =$ (*Write, LoanerFile, WorkingHours, LoanerOffice, ComputerRoom*)
- $WHBackupFile =$ (*Backup, AllFile, WorkingHours, Anywhere, ComputerRoom*)
- $NTBackupFile =$ (*Backup, AllFile, NightTime, Anywhere, ComputerRoom*)
- $NTRestoreFile =$ (*Restore, AllFile, NightTime, Anywhere, ComputerRoom*)

9. $UserAssignment = \{$(*Tom, Teller*), (*Leena, Loaner*), (*Diana, DTSO*), (*Nina, NTSO*), (*Sam, SOM*)$\}$
10. $PermAssign = \{$(*Teller, readTellerFile*), (*Teller, writeTellerFile*), (*Loaner, readLoanerFile*), (*Loaner, writeLoanerFile*), (*DTSO, WHBackupFile*), (*NTSO, NTBackFile*), (*NTSO, NTRestoreFile*)$\}$
11. Role hierarchy $RH = (SOM \succsim_{s,tl} DTSO) \wedge (SOM \succsim_{s,tl} NTSO)$
12. $DSOD_s = (Teller, Auditor)$

This is just one possible way to model the requirements. Other models are possible as well.

9 Conclusion and Future Work

Traditional access control models do not take into account environmental factors before making access decisions. Such models may not be suitable for pervasive computing applications. Towards this end, we proposed a spatio-temporal role based access control model. We identified the entities and relations in RBAC and investigated their dependence on location and time. This dependency necessitates changes in the invariants and the operations of RBAC. The behavior of the model is formalized using constraints.

A lot of work remains to be done. One is the analysis of the model. We have proposed many different constraints. We need to understand the interaction of these constraints and the different types of relationships between them. Specifically, we are interested in finding conflicts and redundancies among the constraint specification. Such analysis is needed before our model can be used for real world applications. We plan to investigate how to automate this analysis. We also plan to implement our model. We need to investigate how to store location and temporal information and how to automatically detect role allocation and enabling using triggers. Once we have an implementation, we validate our model using some prototype application.

Acknowledgement

This work was supported in part by AFOSR under contract number FA9550-07-1-0042.

References

1. Elisa Bertino, Piero Andrea Bonatti, and Elena Ferrari. TRBAC: a temporal role-based access control model. In *RBAC '00: Proceedings of the fifth ACM workshop on Role-based access control*, pages 21–30, New York, NY, USA, 2000. ACM Press.

2. Elisa Bertino, Barbara Catania, Maria Luisa Damiani, and Paolo Perlasca. GEO-RBAC: a spatially aware RBAC. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 29–37, New York, NY, USA, 2005. ACM Press.
3. Suroop Mohan Chandran and J. B. D. Joshi. *LoT-RBAC*: A Location and Time-Based RBAC Model. In *WISE*, pages 361–375, 2005.
4. Michael J. Covington, Prahlad Fogla, Zhiyuan Zhan, and Mustaque Ahamad. A Context-Aware Security Architecture for Emerging Applications. In *Proceedings of the Annual Computer Security Applications Conference* , pages 249–260, Las Vegas, NV, USA, December 2002.
5. Michael J. Covington, Wende Long, Srividhya Srinivasan, Anind Dey, Mustaque Ahamad, and Gregory Abowd. Securing Context-Aware Applications Using Environment Roles. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies*, pages 10–20, Chantilly, VA, USA, May 2001.
6. D.F. Ferraiolo, R. Sandhu, S. Gavrila, and D. R. Kuhn an d R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and Systems Security*, 4(3), August 2001.
7. Urs Hengartner and Peter Steenkiste. Implementing Access Control to People Location Information. In *Proceeding of the SACMAT'04 Yorktown Heights,California,USA*, June 2004.
8. James B.D. Joshi, Elisa Bertino, Usman Latif, and Arif Ghafoor. A Generalized Temporal Role-Based Access Control Model. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):4–23, 2005.
9. Ulf Leonhardt and Jeff Magee. Security Consideration for a Distributed Location Service. *Imperial College of Science, Technology and Medicine, London, UK*, 1997.
10. Indrakshi Ray and Mahendra Kumar. Towards a Location-Based Mandatory Access Control Model. *Computers & Security*, 25(1), February 2006.
11. Indrakshi Ray, Mahendra Kumar, and Lijun Yu. LRBAC: A Location-Aware Role-Based Access Control Model. In *Proceedings of the 2nd International Conference on Information Systems Security*, pages 147–161, Kolkata, India, December 2006.
12. Geetanjali Sampemane, Prasad Naldurg, and Roy H. Campbell. Access Control for Active Spaces. In *Proceedings of the Annual Computer Security Applications Conference* , pages 343–352, Las Vegas, NV, USA, December 2002.
13. Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.
14. Richard Simon and Mary Ellen Zurko. Separation of duty in role-based environments. In *CSFW '97: Proceedings of the 10th Computer Security Foundations Workshop (CSFW '97)*, pages 183–194, Washington, DC, USA, 1997. IEEE Computer Society.
15. Hai Yu and Ee-Peng Lim. LTAM: A Location-Temporal Authorization Model. In *Secure Data Management*, volume 3178 of *Lecture Notes in Computer Science*, pages 172–186, Toronto, Canada, August 2004.