On the Formal Analysis of a Spatio-Temporal Role-Based Access Control Model [⋆]

Manachai Toahchoodee and Indrakshi Ray

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

Abstract. With the growing use of wireless networks and mobile devices, we are moving towards an era where spatial and 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 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. Consequently, researchers have proposed various spatio-temporal access control models that are useful in pervasive computing applications. Such models typically have numerous different features to support the various application requirements. The different features of a spatio-temporal access control model may interact in subtle ways resulting in conflicts. We illustrate how the access control model can be formally analyzed to detect the presence of conflicts. We use Alloy, a formal language based on first-order logic, for the purpose of our analysis. Alloy is supported by a software infrastructure that allows automated analysis of models and has been used to verify industrial applications. The results obtained by analyzing the spatio-temporal access control model will enable the users of the model to make informed decisions.

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 [8] and CMU's Aura [11], 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, access control models and mechanisms that use environmental factors, such as, time and location, while determining access are needed.

Researchers have proposed various access control models that use contextual information, such as, location and time, for performing access control [1, 4–6, 8, 11, 13–17,

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

20] Many of these were developed for commercial applications and are based on RBAC. Examples include TRBAC [4], Geo-RBAC [5], and STRBAC [15]. These models are more expressive than their traditional counterparts, and have various features which the users can selectively use based on the application requirements. The different features of these models interact in subtle ways resulting in inconsistencies and conflicts. Consequently, it is important to analyze and understand these models before they are widely deployed.

Manual analysis is tedious and error-prone. Analyzers based on theorem proving are hard to use, require expertise, and need manual intervention. Model checkers are automated but are limited by the size of the system they can verify. In this paper, we advocate the use of Alloy [12], which supports automated analysis, for checking access control models. Alloy is a modeling language capable of expressing complex structural constraints and behavior. Moreover, it has been successfully used in the modeling and analysis of real-world systems [10, 24].

In this paper, we illustrate how to specify and analyze properties of a spatio-temporal role-based access control model. Alloy is supported by an automated constraint solver called Alloy Analyzer that searches instances of the model to check for satisfaction of system properties. The model is automatically translated into a Boolean expression, which is analyzed by SAT solvers embedded within the Alloy Analyzer. A user-specified scope on the model elements bounds the domain, making it possible to create finite Boolean formulas that can be evaluated by the SAT-solver. When a property does not hold, a counter example is produced that demonstrates how it has been violated.

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

2 Related Work

Ardagna et al. [1] discuss how location information of the requester can be ascertained and how such information can be used to evaluate and enforce location-based predicates used in access control. Location-based access control has been addressed in other works as well [11, 14, 16].

Role-based access control model [9] is used for addressing the access control needs of commercial organizations. Several works exist that improve RBAC functionality, some of which focus on how RBAC can be extended to make it context aware. Sampemane et al. [19] 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. 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. Covington et al. [8] 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. However, the environment roles are activated according to the changing conditions specified in environmental conditions, thus environmental properties like time and location are introduced to the access control framework. In a subsequent work [7], Covington et al. describes the Context-Aware Security Architecture (CASA) which is an implementation of the GR-BAC model.

Other extensions to RBAC include the Temporal Role-Based Access Control Model (TRBAC) proposed by Bertino et al. [4]. 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.[13] 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 [5]. 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. [17]. 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 is not addressed in this paper. None of these work discuss the impact of time on location.

The paper proposed by Chandran et al. [6] combines the main features of GTR-BAC 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. Another work which falls into this category is GST-RBAC by Samuel et al. [20]. In this work, the authors develop a framework to incorporate topological spatial constraints to the existing GTRBAC model. The authors do this by augmenting GTRBAC operations, namely, role enabling, user-role assignment, role-permission assignment, and role-activation with spatial constraints. The operations are allowed only if the spatial and temporal constraints are satisfied. The model also introduces the notion of Spatial Role Hierarchy and Spatial Separation of Duty (spSoD) constraints. Our early work [15] extends RBAC with spatial and temporal constraints. Although the goal of this work is similar to those proposed by Chandran et al. and Samuel et al., our model can express some real-world constraints that are not possible in the other ones. Atluri et al proposed Geotemporal RBAC model in [2, 3] for protecting the information in Geospatial databases. In this model , user can acquire a role based on his spatial-temporal information. For instance, a user can assume the role of a professor in a classroom during the day time. This allows the user to access some resource only when he satisfies spatial and temporal constraints. The Geotemporal RBAC model, however, does not discuss the impact of spatial and temporal constraints on role hierarchy and separation of duties.

A lot of work appears in the area of analysis of security policies. Some have used the Z modeling language for specifying RBAC [25] and LRBAC [17]. Although Z language can represent RBAC and its constraints in the formal manner, the language itself lacks the tool to support the automatic analysis of the formalized model. Others have used an extension of the Unified Modeling Language (UML) [18] called parameterized UML to visualize the properties of RBAC constraints. However, it still lacks the ability to perform automated model analysis.

Researchers have advocated the use of Alloy for modeling and analyzing RBAC specifications. Schaad et al. model user-role assignment, role-permission assignment, role hierarchy, and static separation of duties features of RBAC extension using Alloy in [22]. The authors do not model role activation hierarchy or the dynamic separation of duties. The authors briefly describe how to analyze conflicts in the context of the model. Samuel et al. [20] also illustrate how GST-RBAC can be specified in Alloy. They describe how the various GST-RBAC functionalities, that is, user-role assignment, role-permission assignment, and user-role activation, can be specified by Alloy. However, this work does not focus on how to identify interactions between features that result in conflicts. Our work fills this gap.

3 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[1]. We discuss how the different entities of RBAC, namely, *Users*, *Roles*, *Sessions*, *Permissions*, *Objects* and *Operations*, are associated with location and time.

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, the following constraint must be true. Note that in this and all the subsequent formulae, we omit the quantification symbols.

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

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.

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

[1] For lack of space, we do not give details on location and time representation. Please refer to our earlier paper [15] for more details.

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.

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* [4, 13] 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.

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.

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

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.

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

4 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 [21]. 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. [13] 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.

[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 inherits y's permissions but not the locations and time associated with it. In other words, the permission can be applied wherever the senior role is at that time. This is formalized as follows:

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

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.

[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:

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

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.

[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 inherits y's permissions together with the temporal constraints associated with the permission. This is formalized as follows:

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

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.

[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:

$$(x \succcurlyeq_t y) \wedge UserRoleActivate(u,x,d,l) \wedge d \subseteq RoleEnableDur(y) \Rightarrow$$
$$UserRoleActivate(u,y,d,l')$$

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.

[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:

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

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.

[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:

$$(x \succcurlyeq_l y) \wedge UserRoleActivate(u,x,d,l) \wedge l \subseteq RoleEnableLoc(y) \Rightarrow$$
$$UserRoleActivate(u,y,d',l)$$

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.

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

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

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.

[Time Location Restricted Activation Hierarchy] Let x and y be roles such that $x \succeq_{tl} y$, 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:

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

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.

5 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 [23]. 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.

[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.

[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.

[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.

[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.

[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.

[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.

[Strong Spatial Form of SSoD - Permission Role Assignment] Let p and q be two permissions such that $p \neq q$. $(p,q) \in SSOD\_PRA_l$ 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.

[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.

6 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.
[Weak Form of DSoD] Let x and y be two roles such that $x \neq y$. $(x,y) \in DSOD_w$ 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.
[Strong Temporal Form of DSoD] Let x and y be two roles such that $x \neq y$. $(x,y) \in DSOD_t$ 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.
[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.
[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. An user cannot be a code developer and a code tester in the same session.

7 Model Analysis

An Alloy model consists of *signature* declarations, *fields*, *facts* and *predicates*. Each signature consists of a set of *atoms* which are the basic entities in Alloy. Atoms are *indivisible* (they cannot be divided into smaller parts), *immutable* (their properties do not change) and *uninterpreted* (they do not have any inherent properties). Each field belongs to a signature and represents a relation between two or more signatures. A relation denotes a set of tuples of atoms. Facts are statements that define constraints on the elements of the model. Predicates are parameterized constraints that can be invoked from within facts or other predicates.

The basic types in the access control model, such as, *User*, *Time*, *Location*, *Role*, *Permission* and *Object* are represented as signatures. For instance, the declarations shown below define a set named *User* and a set named *Role* that represents the set of all users and the set of all roles in the system. Inside the *Role* signature body, we have four relations, namely, *RoleAllocLoc*, *RoleAllocDur*, *RoleEnableLoc*, and *RoleEnableDur* which relates *Role* to other signatures.

```
sig User{}
sig Role{
 RoleAllocLoc: Location,
 RoleAllocDur: Time,
 RoleEnableLoc: Location,
 RoleEnableDur: Time}
```

The different relationships between the STRBAC components are also expressed as signatures. For instance, *RoleEnable* has a field called *member* that maps to a cartesian product of *Role*, *Time* and *Location*. Similarly, *RoleHierarchy* has a field *RHmember* that represents a relationship between *Role* and *Role*. Different types of role hierarchy are modeled as the subsignatures of RoleHierarchy.

```
sig RoleEnable { member: Role -> Time -> Location}
sig RoleHierarchy { RHmember: Role -> Role}
sig UPIH, TPIH, LPIH, TLPIH, UAH, TAH, LAH, TLAH extends
     RoleHierarchy{}
```

The various invariants in the STRBAC model are represented as facts in Alloy. For instance, the fact *URActivate* states that for user *u* to activate role *r* during the time interval *d* and location *l*, this user has to be assigned to role *r* in location *l* during time *d*. Moreover, the location of the user must be a subset of the locations where the role is enabled, and the time must be in the time interval when role *r* can be enabled. This is specified in Alloy as shown below. Other invariants are modeled in a similar manner.

```
fact URActivate{
all u: User, r: Role, d: Time, l: Location, uras: UserRoleAssignment,
urac: UserRoleActivate |
((u->r->d->l) in urac.member) => (((u->r->d->l) in uras.member) &&
(l in r.RoleEnableLoc) && (d in r.RoleEnableDur))
}
```

To represent the effects of STRBAC hierarchical structure, we use Alloy's *fact* feature. The fact *UPIHFact* represents the Unrestricted Permission Inheritance Hierarchy's property. The fact states that senior role `sr` can acquire all permission assigned to itself together with all permissions assigned to junior role `jr`.

```
//Unrestricted Permission Inheritance Hierarchy
fact UPIHFact{
     all sr, jr: Role, p: Permission, d: Time, l: Location, upih: UPIH,
         rpa: RolePermissionAssignment, pra: PermRoleAcquire |
         ((sr->jr in upih.member) && (jr->p->d->l in pra.member) &&
         (sr->p !in (rpa.member).Location.Time)) =>
              (sr->p->sr.RoleEnableDur->sr.RoleEnableLoc) in pra.member}
```

The separation of duty constraints are modeled as predicates. Consider the Weak form of Static Separation of Duties User Role Assignment. This constraint says that a user *u* assigned to role *r*1 during time *d* and location *l* cannot be assigned to its conflicts role *r*2 at the same time and location. The other forms are modeled in a separate manner.

```
//Weak Form of SSoD-User Role Assignment
pred W_SSoD_URA(u: User, disj r1, r2: Role,
ura: UserRoleAssignment.member, d: Time, l: Location){
((u->r1->d->l) in ura) => ((u->r2->d->l) not in ura)
}
```

Finally, we need to verify whether any conflicts occur between the features of the model. We rely on the powerful analysis capability of the ALLOY analyzer for this purpose. We create an *assertion* that specifies the properties we want to check. After we create the assertion, we will let ALLOY analyzer validate the assertion by using *check* command. If our assertion is wrong in the specified scope, ALLOY analyzer will show the counterexample.

For instance, to check the interaction of the Weak form of SSOD User Role Assignment and the Unrestricted Permission Inheritance Hierarchy, we make the assertion shown below. The assertion does not hold as illustrated by the counterexample shown in Figure 1.

```
// WSSoD_URA violation in the present of UPIH Hierarchy
check TestWSSoD_URA
assert TestConflict1_1{
no u: User, disj x, y: Role, upih: UPIH,
      d: Time, l: Location, ura: UserRoleAssignment |
((x->y in ^(upih.member))  &&
            (u->x->d->l in ura.member)) =>
W_SSoD_URA[u, x, y, u->(x+y)->d->l, d, l]
}
check TestConflict1_1
```

The counterexample shows one possible scenario. In this case, it uses the following instances to show the violation.

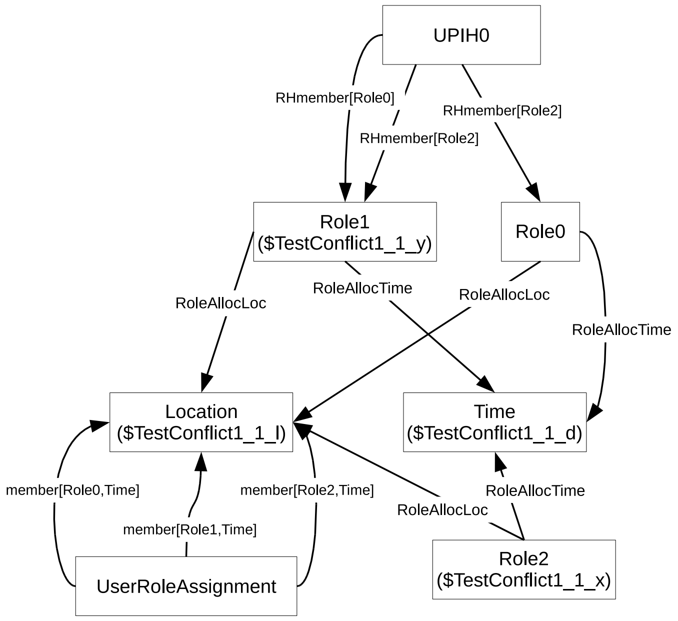

Fig. 1. Counterexample for assertion TestConflict1_1

1. $Role = \{Role0, Role1, Role2\}$
2. $UPIH0 = \{Role0 \rightarrow Role1, Role2 \rightarrow Role0, Role2 \rightarrow Role1\}$
3. $Time = d$, $Location = l$
4. $UserRoleAssignment = \{User \rightarrow Role0 \rightarrow Time \rightarrow Location, User \rightarrow Role1 \rightarrow Time \rightarrow Location, User \rightarrow Role2 \rightarrow Time \rightarrow Location\}$

Substituting x and y in W_SSoD_URA predicate with $Role2$ and $Role1$ respectively, we get the violation. We checked the assertion on a HP-xw4400-Core2Duo-SATA with two Core2Duo 1.86Ghz CPU and 2 Gb memory running Linux 64. We used Version 4.1.2 Alloy Analyzer. The time taken to check this assertion was 25,916 ms.

Similar types of analysis reveals that the various forms of SSoD permission role inheritance conflict with the different forms of permission inheritance hierarchy. Conflicts were also detected with the various forms of SSoD user role assignment with different forms of permission inheritance hierarchy. Also, the various forms of DSoD constraints conflict with the different forms of role activation hierarchy. Another source of conflict occurs between role activation and permission when the corresponding location constraints or the temporal constraints do not overlap. Checking all these conflicts took 914,558 ms in our setup.

8 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. We investigated how the different constraints interact with each other and their relationships.

A lot of work remains to be done. We also plan to implement our model. We need to investigate how to store location and temporal information in an optimal manner. We also need to investigate how to use triggers for automatically detecting role allocation and role enabling. Once we have an implementation, we plan to validate our model using some example real-world applications. We also plan to adapt our analysis techniques for verifying other types of access control models.

References

1. Claudio A. Ardagna, Marco Cremonini, Ernesto Damiani, Sabrina De Capitani di Vimercati, and Pierangela Samarati. Supporting location-based conditions in access control policies. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security*, pages 212–222, Taipei, Taiwan, March 2006.
2. V. Atluri and Soon Ae Chun. An authorization model for geospatial data. *IEEE Transactions on Dependable and Secure Computing*, 1(4):238–254, October-December 2004.
3. Vijayalakshmi Atluri and Soon Ae Chun. A geotemporal role-based authorisation system. *International Journal of Information and Computer Security*, 1(1/2):143–168, January 2007.
4. Elisa Bertino, Piero Andrea Bonatti, and Elena Ferrari. TRBAC: a temporal role-based access control model. In *Proceedings of the 5th ACM Workshop on Role-Based Access Control*, pages 21–30, Berlin, Germany, July 2000.
5. Elisa Bertino, Barbara Catania, Maria Luisa Damiani, and Paolo Perlasca. GEO-RBAC: a spatially aware RBAC. In *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies*, pages 29–37, Stockholm, Sweden, June 2005.
6. Suroop Mohan Chandran and J. B. D. Joshi. *LoT-RBAC*: A Location and Time-Based RBAC Model. In *Proceedings of the 6th International Conference on Web Information Systems Engineering*, pages 361–375, New York, NY, USA, November 2005.
7. 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.
8. 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.
9. 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):224 – 274, August 2001.

10. Geri Georg, James Bieman, and Robert B. France. Using Alloy and UML/OCL to Specify Run-Time Configurati on Management: A Case Study. In Andy Evans, Robert France, Ana Moreira, and Bernhard Rumpe, editors, *Practical UML-Based Rigorous Development Methods - Countering o r Integrating the eXtremists.*, volume P-7 of *LNI*, pages 128–141. German Informatics Society, 2001.
11. Urs Hengartner and Peter Steenkiste. Implementing Access Control to People Location Information. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies*, pages 11–20, Yorktown Heights, NY, USA, June 2004.
12. Daniel Jackson. *Alloy 3.0 reference manual*. At http://alloy.mit.edu/reference-manual.pdf, 2004.
13. 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, January 2005.
14. Ulf Leonhardt and Jeff Magee. Security Consideration for a Distributed Location Service. *Imperial College of Science, Technology and Medicine, London, UK*, 1997.
15. I. Ray and M. Toahchoodee. A Spatio-temporal Role-Based Access Control Model. In *Proceedings of the 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pages 211–226, Redondo Beach, CA, July 2007.
16. Indrakshi Ray and Mahendra Kumar. Towards a Location-Based Mandatory Access Control Model. *Computers & Security*, 25(1), February 2006.
17. 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.
18. Indrakshi Ray, Na Li, Robert France, and Dae-Kyoo Kim. Using UML to Visualize Role-Based Access Control Constraints. In *Proceedings of the 9th ACM symposium on Access Control Models and Technologies*, pages 115–124, Yorktown Heights, NY, USA, June 2004.
19. 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.
20. Arjmand Samuel, Arif Ghafoor, and Elisa Bertino. A Framework for Specification and Verification of Generalized Spatio-Temporal Role Based Access Control Model. Technical report, Purdue University, February 2007. CERIAS TR 2007-08.
21. Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.
22. Andreas Schaad and Jonathan D. Moffett. A Lightweight Approach to Specification and Analysis of Role-Based Access Control Extensions. In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies*, pages 13–22, Monterey, CA, USA, June 2002.
23. Richard Simon and Mary Ellen Zurko. Separation of Duty in Role-based Environments. In *Proceedings of the 10th Computer Security Foundations Workshop*, pages 183–194, Rockport, MA, USA, June 1997.
24. Mana Taghdiri and Daniel Jackson. A lightweight formal analysis of a multicast key management scheme. In *Formal Techniques for Networked and Distributed Systems - FORTE 2 003*, volume 2767 of *Lecture Notes in Computer Science*, pages 240–256. Springer, 2003.
25. Chunyang Yuan, Yeping He, Jianbo He, and Zhouyi Zhou. A Verifiable Formal Specification for RBAC Model with Constraints of Separation of Duty. In *Proceedings of the 2nd SKLOIS Conference on Information Security and Cryptology*, pages 196–210, Beijing, China, November 2006.