Accessible Spatio-Temporal Role-Based Access Control

Mahendra Kumar and Richard E. Newman
Department of Computer & Information Sciences & Engineering
University of Florida
Gainesville, FL, USA, 32611
{makumar, nemo}@cise.ufl.edu

June 14, 2011

Abstract

The rapid emergence of GPS enabled devices, sensors, and mobile equipment in commercial as well as government organizations has led to considerable research in location-based access control schemes. This paper introduces the concept of time- and location-based access control, and shows how role-based access control (RBAC) model can be extended to incorporate these parameters for granting access. Location based access policies enhance the security of an application by restricting access to an object only from specified locations. On the other hand, temporal constraints limit access to a specific time interval. This paper presents an understandable user interface for a spatio-temporal RBAC (STRBAC) system we are developing. It shows both the complexity of the decisions that must be made, and the relative understandability of the system for users and for administrators who must configure the system. It is critical not only to be able to express and enforce desired policies, but also to determine what the current policies are. This interface provides an accessible mechanism for this form of access control.


1 Introduction

Time and location are one of the important attributes that determine the access control properties in an organization. Such constraints are essential for controlling various location and time sensitive applications in organizations. While location constraints limit the access of resources from a particular set of pre-defined locations, temporal constraints put restrictions on the time-limit for access of these resources. Consider the following situations. The computer science department at our local university might require that students in the department should be able to avail the online resources viz. printers, fax machines and a particular group of file servers only if they are present inside the department building and that too during regular office hours (9am to 5pm). A similar scenario could be that the chief manager of a bank should be able to access very important electronic files if and only if he is present inside a particular room and during normal banking hours. It is clear that in the above scenarios we require the access policies to be defined in such a manner that it depends not only on the role of the individual but also on the location as well as the time during which the resource can be accessed.

We have provided a practical implementation of the above mentioned model by developing a simulator which provides access to a resource based on the role, location and time. Our system provides three graphical user interfaces (GUI) which enable the system administrator to define access policies in a convenient manner that is simple and easy to interpret. We introduce the concepts of logical time and location. We also show that logical location and time have a hierarchical structure that can be used to refine accessibility issues.
This paper is organized as follows. In Section 2, we provide some background and related work. Section 2.1 summarizes the RBAC model on which our work is based while Section 2.2 gives information about different devices which can be used to determine the location of an entity. Section 3 illustrates details about spatio-temporal policy specification and their implementation. Finally, Section 4 concludes the paper with reference to future pointers.

2 Background

2.1 Role-Based Access Control

Role based access control (RBAC) [5] has emerged as one of the promising alternative to the traditional discretionary and mandatory access control techniques. In recent years, RBAC has received considerable attention and recognition because of its flexibility and ease in defining access policies within an organization. The main components of RBAC consist of a set of users (\textit{USERS}), a set of roles (\textit{ROLES}), a set of objects (\textit{OBJs}), a set of operations (\textit{OPS}) and a set of permissions (\textit{PERMISSIONS}). Roles are assigned to different users while a set of permissions are assigned to different roles. Hence, when a role is assigned to a particular user, the user automatically acquires all the permissions associated with that specific role. Since the roles directly represent job functions in an organization, we do not need to define separate permissions for every user in an organization, thus making it easier to implement access policies when the number of users is very large. The assignment of roles greatly simplifies the security policy management within an organization. The core Role-based Access control model is represented in figure 1. The multiple relationship between different components of the RBAC Model is represented by arrowheads. For example, one role can be assigned to many users and one user can be associated with many roles. Hence a double arrowhead is shown on both \textit{USERS} and \textit{ROLES}.

2.2 Location Determination

To determine the location of a subject/object, we need devices which can provide this information to the location server. GPS is one of the most popular and inexpensive methods to determine location information. Although GPS based devices provide a wide coverage area, their accuracy is limited. Infrared sensors are another type of devices which can determine the location of a user with high accuracy. In such a location-determining system, the infra-red base sensors occupy fixed positions within a building. All the sensors are interconnected to each other by means of a high-speed communication network. Every object has built-in infra-red transponders and the users are required to wear active badges which can respond to the sensors [8]. This type of system is ideally suited for small organizations. The other type of location-determining systems can be a base station which can predict that a user is within a certain geographical area. For example, in a wireless network, a particular base station can determine the location of the mobile users if those users are within its sensing range. Other types of sensors in use these days include RF-IDs, speech recognition and image processing devices.

Various methods of location representation and determination have been discussed in [2, 6, 7, 10, 11]. Hengartner and Steenkiste [9] proposed a system for locating people at different locations. They show how location information can be verified using secure methods called service trust. They have also pointed out the need for location granularity and time constraints.

Many real world examples showing the need for location-based authentication have been mentioned by Denning and MacDoran [3]. They have suggested the use of GPS devices for obtaining location information. They have also discussed about CyberLocator, a location-based authentication tool. Harter and Hooper [8] modeled a location system for an active office where they applied the use of infra-red devices for obtaining the location of people. Each room has an infra-red sensor and the people wear infra-red transponders called Active Badges. Lee, Xu, Zheng and Lee [4]
discussed several methods to represent location. They consider location to have a hierarchical structure and propose two different models, viz. the Geometrical model and the Symbolic model.

3 Policy Specification and Implementation

To implement the STRBAC [12] policies, we first need to define some new terms which we will be using in our implementation.

3.1 Physical Location

We already discussed various ways to determine the physical location of subjects/objects in an access control system. In our model, we assume that we have a location sever which maps every sensor to its physical location. When a user is in a particular location, the sensor in that location picks up the signal from the user’s transmitting device and sends an interrupt to the location server. The location server in turn, finds out the mapping of the sensor to the physical locations and stores the user’s location in a pre-specified directory.

Every physical location has a parent location except the root location which we call as BASE. BASE can initially be defined or set by the administrator. For example, if the STRBAC system has to be implemented for the University of Florida then we can set the BASE to be the University of Florida. All other physical locations are direct or indirect children of this BASE. Hence, physical location has a hierarchical structure with BASE as the root on top of the hierarchy. As an example, the Computer Science building is an immediate child of the University of Florida. Further, the first floor inside this building is an immediate child of the Computer Science building. In some cases, two physical locations may overlap with each other but for simplicity we do not consider that issue in this paper and leave it for future work. Figure 2 below shows an example of such a location hierarchy where BASE is the University of Florida.

3.2 Logical Location Specification

Logical location helps in naming physical locations in a way which is more suitable for human understanding. It also helps in combining a set of physical or logical locations into a single unit and hence reduces the overhead of defining the locations constraints individually. For example, it is more appropriate for the humans to call a set of rooms as lecture rooms or a particular room as the department office. We will see how logical location helps the system administrators to define policies in a much more convenient way. Logical location can be further categorized into three types: (1) Location-by-Use, (2) Location-by-Organization and (3) User-defined locations.

In order to associate location with its usage, we introduce the concept of Location-by-Use. For example, we may designate a set of rooms as classrooms. Now, if we want to specify policies where we want to give access to all the students of the department when they are inside any classroom, we can simply select classrooms as the location and it will implicitly include all the rooms which
are designated as classrooms. The advantage of defining location in this way is that we do not have to select every location individually each time we need to define a policy that involves a set of locations.

Location-by-Organization is helpful in defining locations when they can be distinguished based on the organizational level. For example, it might be that more than one department/organization may be co-located at the same building. For example, the Computer Science Department and the University Computing Center are located within the same building. The first four floors are under the jurisdiction of Computer Science whereas the fifth floor is under the University Computing Center. Hence, for defining and implementing policies based on organization, we need the locations categorized by organizations.

User-defined locations are used to create custom locations which do not fall in any logical category. This is helpful if the administrator wants to create some shortcut to a set of locations. For example, the Security Research group in our department does not fall under any logical location type and is comprised of a set of rooms and a lab. Hence, if we want to define this type of logical location, we can place it under user-defined location types.

As discussed before, physical location implicitly has a hierarchical structure. But, Location-by-Use and Location-by-Organization can also have a hierarchical structure. This is shown by Figure 2 which shows the relationship among the different type of locations.

A location constraint is a group of logical locations joined by the OR or the MINUS operator.

**Definition 1 [Location Constraint representation]** A location constraint $L_s$ is a combination of one or more logical or physical locations joined by an OR or/and MINUS operator. Thus,

$$L_s = L_{s1} \star L_{s2} \cdots L_{sn}$$  \hspace{1cm} (1)

where operator $\star \in \{\lor, -\}$ and $L_s$ is a logical or physical location.
The OR operator specifies that the role member can access the object if he is in one of the locations. The MINUS operator specifies negative constraint which means that the user is not allowed access from the location associated with minus operator. In our GUI the MINUS operator is represented by NOT.

3.3 Logical Location Implementation Module

For adding new logical locations to the existing location tree, we developed a GUI tool which helps the administrator to select the appropriate type of existing locations from the tree along with operators which define the new location. Our GUI screen shot shown in Figure 3 illustrates this by adding a new location called Psychology Department to the existing location tree. The steps required to add a new logical location to the existing location tree in our GUI is explained below.

Step 1 requires the user to input the new logical location name which he wants to add. In step 2, the immediate parent location is selected under whom the new location will be displayed. When the user selects the parent location, it is selected in white color. The GUI in Figure 3 shows that we wish to add a new location called Psychology Department under the parent location – College of Liberal Arts & Sciences. Step 3 helps in selecting the location units and adding it to the third panel by ADD button. The third panel shows the list of all the currently selected location which comprise this new logical location. We can associate several locations with common operators such as

Figure 3: The screen shot of the GUI developed to define location
as (,), AND, OR and NOT to form a single complex location. Common operators can be selected from the middle panel shown by the light grey color. In Figure 3, the expression in the third panel means that the Psychology department consists of all the physical location in the Psychology building except the third floor. The NOT operator is used to exclude a certain location. Observe that we have selected all the physical locations in the Psychology Building except the third floor. The NOT operator is used to indicate that the third floor of Psychology building is not under the Psychology Department. This situation may arise when the same building may host more than one department/organization.

3.4 Time Interval Representation

A very important part of STRBAC also deals with temporal constraints. Hence, we need to define time in a manner such that it can be used to represent any recurring or non-recurring time interval and can be used in temporal expressions. Recurring interval are used for defining access policies which occur periodically after a certain interval of time. Non-recurring interval specify the start-date and the end-date of a time interval.

1. **Non Recurring Interval:** We use yyyy/mm/dd format for representing date in STRBAC system. A Non Recurring Interval in STRBAC system is represented by \(\Delta = (\delta_s, \delta_e)\), where \(\delta_s\) and \(\delta_e\) denote the start date and end date between which the access is valid.

2. **Recurring Interval:** Recurring Interval are used to define policies where access is granted in terms of weekly, daily or monthly basis. Consider an example where we want that a particular operation should be allowed only on Mondays and Fridays of every week or the 10\(^{th}\) day of each month or the 100\(^{th}\) day of each year. As another example, we may need to define access policies in a manner such that an operation should be allowed only between 10 AM to 12 PM on Mondays of every week or 10\(^{th}\) day of every month. Salary slips should only be printed on the last working day of each month and not any other day. Employees in a company should be able to avail the services of the printer only during the office hours of working days and not on weekends or during off hours. All the above examples represent Recurring Interval. For these situations, we need an access policy which can recognize these recurring intervals. We represent these interval in a manner similar to Bertino, Bettini, Ferrari and Samarati [1]. Recurring Interval are sub-divided into four categories: Daily, Weekly, Monthly and Yearly.

   - **Daily Interval** denotes the time duration in a day during which the access is granted. We use HH : MM : SS format to represent time which is a 24 hour time representation format to denote hour, minute and second. For our paper we assume that every entity in STRBAC is synchronized to a global clock which is independent of any geographical location. A Daily Interval denoted by \(\Gamma_D\), is represented as \((\gamma_s, \gamma_e)\), where \(\gamma_s\) and \(\gamma_e\) represent the start-time and end-time respectively. If \(\Gamma_D = (\infty, \infty)\), then we assume that access is granted for 24 hours a day.

   - **Weekly Interval** denotes the daily periodicity in a week during which the access is granted. We denote this constraint as \(\Gamma_W\), and represent it as \(\{x_1 \cdots x_n\}.day.week\) where \(1 \leq x_i \leq 7\). \(x_i\) signifies the day of week during which access is granted. Sunday is assumed to be the first day of every week. Hence, \(\{2, 4\}.day.week\) represents that access is granted only on Mondays and Wednesdays of every week. If \(\Gamma_W = \{\infty\}.day.week\), we assume that access is valid for every day of the week.

   - **Monthly Interval** denotes the daily or weekly periodicity in a month for which access is granted. It consists of two different types of Interval :day Interval in a month and the week Interval in a month. Day Interval in a month is denoted by \(\Gamma_{Md}\) and is represented
as \( \{x_1 \cdots x_n\}.\text{day.month} \) where \( 1 \leq x_i \leq ldm \). \( ldm \) represents the number of days in a month and can be set by simple calculations. \( x_i \) denotes the day of the month on which the access is valid. Hence, \( \{2,14\}.\text{day.month} \) represents that access is valid only on the \( 2^{nd} \) and \( 14^{th} \) day of each month. Similarly, week Interval in a month is denoted by \( \Gamma_{mw} \) and is represented as \( \{x_1 \cdots x_n\}.\text{week.month} \) where \( 1 \leq x_i \leq 4 \) or \( x_i = lwm \). Here, \( x_i \) denotes the week of the month during which the access is valid. Hence, \( \{2,4\}.\text{week.month} \) represents that access is granted only during the \( 2^{nd} \) and \( 4^{th} \) week of each month. When \( x_i = lwm \), then we assume that access is granted for the last week of the month i.e. the last seven days of the month.

- **Yearly Interval** denotes the daily, weekly or monthly periodicity in a year during which access is valid. Daily interval in a year denoted by \( \Gamma_{yd} \), is represented as \( \{x_1 \cdots x_n\}.\text{day.year} \) where \( 1 \leq x_i \leq ldy \). \( ldy \) represents the number of days in a year and can be set by the system automatically by simple calculations. \( x_i \) denotes the day of the year on which the access is valid. Hence, \( \{50,100\}.\text{day.year} \) signifies that access is granted only on the \( 50^{th} \) and \( 100^{th} \) day of each year. Weekly interval in a year is denoted by \( \Gamma_{yw} \) and is represented as \( \{x_1 \cdots x_n\}.\text{week.year} \) where \( 1 \leq x_i \leq 52 \). Here, \( x_i \) implies the validity of access rights only on the \( x_i^{th} \) week of each year. Hence, \( \{2,34\}.\text{week.year} \) denotes that access is granted only during the \( 2^{nd} \) and \( 34^{th} \) week of each year. Similarly, Monthly interval in a year is denoted by \( \Gamma_{ym} \) and is represented as \( \{x_1 \cdots x_n\}.\text{month.year} \) where \( 1 \leq x_i \leq 12 \). \( x_i \) denotes the month in a year during which access is valid. Hence, \( \{2,4\}.\text{month.year} \) indicates that access is granted only during February and April of each year. Again \( \{\infty, \infty\}.\text{day.year}, \{\infty, \infty\}.\text{week.year} \) or \( \{\infty, \infty\}.\text{month.year} \) means that the access is granted every day, every week or every month of the year respectively.

Figure 4 summarizes the different types of time interval along with examples.

**Definition 2 [Temporal Expression]** A temporal expression \( P_t \) is an expression combining one or more time intervals joined by an AND, OR or MINUS operator. Thus,

\[
P_t = C_1 \star C_2 \star C_3 \cdots C_n
\]

where interval type \( C_i \in \{\Delta, \Gamma_D, \Gamma_W, \Gamma_{Md}, \Gamma_{Mw}, \Gamma_{Yd}, \Gamma_{Yw}, \Gamma_{Ym}\} \) and the operator \( \star \in \{\land, \lor, -\} \).

The figure 5 shows the different categories of time.
3.5 Logical Time

Specifying temporal constraint in terms of the temporal expression is complicated, consumes too much time and is not user friendly. We therefore define the concept of logical time where a temporal expression is given a user friendly name. For example, the time interval 8 AM to 4 PM between Monday to Friday can be given the logical name of “office hours”. Instead of specifying the whole temporal expression each time, we can define this new logical time “office hours” which in turn can be used again and again in a very easy manner. Now whenever we want to specify the above temporal constraint, we can simply specify it by “Office hours”. Another example can be the last working day of every month which can be given the logical name of “Salary Day”. Since, Salary day is defined as the last working day of month, we can make salary day as a child of Office hours. Similarly, list of holidays can include Federal holidays and Academic holidays. From the above examples, we can infer that logical time has a hierarchical structure. Figure 5 shows a typical example of logical time hierarchy.

3.6 Logical Time Implementation Module

We provide a graphical user interface (GUI) which helps in mapping set of temporal expressions to a new logical names. These logical names in turn can be combined with other existing logical names or the temporal expressions to specify other logical names. This helps in defining and using complex and frequently used temporal constraints in a very convenient way. Figure 5 shows the snapshot of the GUI module which we built to define and add new logical time.

In the GUI in Figure 7, step 1 is used to specify the name of the new logical time which we want to create. In this case we want to define a new logical time called Federal Holidays. Step 2 helps in selecting the immediate parent node under whom the new logical time will be displayed. Since we are defining this holiday for the University of Florida, hence we select Holiday List under University of Florida as the parent node from the first panel. The selected parent node is marked in white color after the user selects it. Step 3 helps the user to select logical time units from the middle panel and/or temporal expression from the bottom which define this new logical time and add it to the rightmost panel. Since we want to define the Federal Holidays, we choose a list of dates which fall under this holiday. These are shown in numbers which means that any of these dates will be considered as Federal holidays. The check boxes are used to define which of the temporal expression we want to select to specify the constraints. If we select more than one temporal expression at one time then \( AND \) or \( OR \) operators are used to combine these expressions. If we select only one expression or logical unit at a time then we do not need any operator. Beginning or End in front of the temporal Expressions is used to specify whether to count the days or weeks from beginning.
or end. For example, to specify the last week of a month, we can specify 1 thru 1 in Week of the Month Option. In this way the system will consider 1 as the 1st week of the month from end of the month. Similar is the case for the other options. The Select Week Type is used to choose the type of week interval we want to choose. For example the 1st partial week in month can consist of a Sunday only if the week is defined as Monday to Sunday and the 1st day of the month falls on Sunday. The other options asks for Week Type which is used to define a week. In calendars a week can either go from Sunday through Saturday or it can go from Monday through Sunday. The Second panel also consists of Common Operators marked in red text. These operators are used to define complex expressions comprising of many temporal and logical units.

3.7 Defining Policies

Our application presents a user friendly GUI to set the desired spatio-temporal policies. The interface is very convenient for the system administrators to specify STRBAC policies in an easy way.

Step 1 enables the administrator to specify a name for that access policy. Since, in this case we are defining a policy which controls the access to labs and lecture room for graduate students, we can name it as Graduate Student access. It depends upon the naming convention what the administrator wants to use for its application. Step 2 helps in selecting the specific roles for which we are implementing the policy. In this case we selected two roles, Masters and Doctorate which appear as 1 and 2 in the white text box in Roles category. This box called the rule box shows the list of all the selections in their corresponding categories and define a rule. Two roles can be selected and can be joined by a AND, OR or NOT operator. For example, consider a scenario where we want to give unlimited printing access to all the Grad students who are either Teaching assistants, research assistants or Fellowship but not to the grad students who are not assistants or
on fellowship. In this case we have to define the role as the following : (Masters OR Doctorate) AND (Teaching Assistant OR Research Assistant OR Fellowship). In the GUI, we use numbers for the roles Masters and Doctorate. This implicitly means that the roles are joined by the OR operator. Step 3 helps in selecting all the locations from where this access policy is valid. In the GUI illustration we have the CISE department as the required location. Step 4 is used to specify the temporal constraint on the access to the object. In the example we want that the unlimited printing facility should be available only during the office hours. Hence, the office hours appear as in the Time column in the Rule box. Steps 5 and 6 are used to specify the object and the type of permission which we want to specify. As a whole, in the above GUI we have defined a rule which says that only the Masters and Doctorate students who are either Teaching or Research assistants or fellowship students have unlimited access to printing facility if they are inside the CISE department and the access too is granted only during office hours.

4 Conclusion

The work described in this paper focuses on specifying policies for spatio-temporal access control using the existing RBAC model. We have formalized the concept of spatio-temporal constraints in the form of boolean expressions. We have also shown in our model how different components of RBAC are related to location and time. This model is useful in the field of pervasive computing where both time and location of the user will be important parameters in determining access to an object. In future, we would like to consider implementation issues and performance as well as issues related to the semantics of spatio-temporal constraints.
References


