Policy Mapper: A Simplified Approach for Administration of Location-based Access Control Policies

Rafae Bhatti
Purdue University, USA
rafa@ieee.org

Maria Luisa Damiani
Università di Milano, I
damiani@dico.unimi.it

David W. Bettis
Purdue University, USA
dwbettis@cs.purdue.edu

Elisa Bertino
Purdue University, USA
bertino@cerias.purdue.edu

February 12, 2007

Abstract

An important issue in mobile computing systems is the administration of location-based access control policies, particularly the mechanism for specification and enforcement of spatial constraints. Simplifying the administration of such policies requires a mechanism that supports both an intuitive and scalable spatial constraint specification and a flexible enforcement architecture. In this paper, we first motivate the significance of these two requirements, and then present Policy Mapper, an administrative tool that is aimed at simplifying the administration of location-based access control policies. The premise of Policy Mapper is based on defining access control at a conceptual level and a logical level to carry out the tasks of constraint specification and enforcement, respectively, and designing an interface definition language to allow the coupling of the two levels. The development of Policy Mapper allows us to bridge a critical gap with regards to expressiveness and enforcement of spatial constraints in location-based access control policies.

Keywords: Access Control, Location-based Constraints, Policy Administration, Mobile Computing

1 Introduction

The widespread deployment of location-based information systems and services as well as increasing concern for management and sharing of geographical information in strategic applications has led to a renewed interest in research in location-based security. While RFID and related technologies for location-based authentication have advanced this field, a comprehensive solution to location-based security should also include mechanisms for administration of location-based access control policies, which
are characterized by requirements for specification and enforcement of spatial con-
straints.

1.1 Motivating Example

Consider a healthcare information service providing emergency medical care to daily
commuters on a local highway. Suppose that the set of users includes different cate-
gories of care-givers, such as nurses, doctors and pharmacists. Each category of care-
givers needs to access different information resources depending on the emergency
context. Moreover, since the emergency medical staff is not physically bound to a ded-
cicated hospital when on duty, the assignment of privileges depends on the location of
the emergency they are attending to.

For example doctors should be allowed to get detailed medical information about
a commuter from all the hospitals along certain sectors of the highway where the ac-
cident occurred. Nurses, on the other hand, should be able to access only the medical
summary, whereas pharmacists can only access the prescription history of the com-
muter. In this scenario, both the hospital and highway sector information refer to well
defined locations in the reference space. In this context, we may need to specify for
example that a doctor could be allowed to get information for a commuter involved in
an accident only when it is confirmed that the mobile unit from which the doctor is
operating is within a certain sector of the highway.

1.2 Problem Statement

We observe from the preceding example that administration of location-based access
control policies has two distinct requirements, namely specification and enforcement
of spatial constraints, which must be addressed together for the design of a location-
based access control scheme. Simplified administration, moreover, requires that the
specification:

- be scalable in order to support the encoding of constraints for various categories
  of users (such as doctors, nurses and pharmacists) as opposed to per-user con-
  straints which could lead an explosion in size of the policy for a huge mobile
  user community, and

- be intuitive enough to allow policies to be constructed in a convenient and user-
  friendly manner, so as to offset the increase in complexity of policies that ac-
  company more expressive power, and

- be expressible in a language that is understood by an underlying enforcement
  mechanism.

On the hand, simplified administration also requires that the enforcement mechanism:

- be flexible so that it can be suitably extended to accept a language that can match
  the expressivity of the constraint specification.
These requirements address two different aspects of access control, namely conceptual level and logical level. While access control at conceptual level enhances policy administrator experience, access control at logical level enables enforcement by ensuring that the expressive spatial constraints can be implemented on the target system. Table 1 summarizes the requirements for simplified policy administration at the conceptual and logical levels of access control, as well as the interface of the two levels.

<table>
<thead>
<tr>
<th>Access Control Level</th>
<th>Requirement</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual (Specification)</td>
<td>Scalable</td>
<td>Accommodate User Categories</td>
</tr>
<tr>
<td></td>
<td>Intuitive</td>
<td>Facilitate User-Friendly Policy Construction</td>
</tr>
<tr>
<td></td>
<td>Reducible</td>
<td>Define Language to Allow Enforcement</td>
</tr>
<tr>
<td>Logical (Enforcement)</td>
<td>Flexible</td>
<td>Provide Extensions to Accept Language</td>
</tr>
</tbody>
</table>

Table 1: Requirements for simplified policy administration

1.3 Proposed Solution

In this paper, we present an administrative tool, called Policy Mapper, that is aimed at defining access control at a conceptual level and a logical level to carry out the tasks of constraint specification and enforcement, respectively, and designing an interface definition language to allow the coupling of the two levels, with the overall goal of simplified policy administration. The conceptual level in our scheme is based on the spatial constraint vocabulary of an existing location-based access control model, GeoRBAC [1], whereas the logical level is based on a well-known context-aware policy specification language, X-GTRBAC [2]. We indicate how GEO-RBAC and X-GTRBAC can be used to meet the requirements for the conceptual and logical levels of access control, respectively, to help achieve our goal of simplified administration. We also define a formal mapping between administrative commands of GEO-RBAC and X-GTRBAC which is used to design an interface definition language. Both design-time and run-time administrative commands are translated between the conceptual and logical levels of access control using this language. The proposed approach is shown in Figure 1.

The remainder of the paper is structured as follows. We first discuss the state of the art in location-based access control with respect to the requirements outlined in Table 1, and summarize our contributions. We then provide an overview of our proposed approach. The following section introduces the GEO-RBAC access control model and the X-GTRBAC policy specification language. We then describe in detail the architectural framework of Policy Mapper. Final remarks and future plans conclude the paper.

We note that in this paper we focus only on the constraint specification and enforcement in location-based access control. We assume that the location information has already been obtained and verified by the location-acquisition and verification technologies, and do not deal with those mechanisms.
2 Background and Related Work

The use of context-aware access control policies with focus on spatial constraints has recently gained a lot of attention [3, 4, 5, 6, 7, 8, 9, 10]. Work on spatial access control models based on role-based access control (RBAC) [11] has been reported in [4, 5, 6]. Since RBAC meets the requirement of allowing constraint specification based on user roles or categories, we believe RBAC is well-suited to act as the baseline for location-based access control in mobile computing applications [1]. However, none of these approaches provide support for a tool (such as GUI) to allow user-friendly construction of the often complicated policies that are a result of adding spatial extensions to the RBAC model. Nor do they provide a mechanism to reduce location-based policies into a language that can be understood by an enforcement mechanism. Therefore, none of these approaches meet the requirements for conceptual level of access control as outlined in Table 1.

The approaches presented in [7, 8, 9, 10] provide support for enforcement of context-aware access control policies. These systems, however, accept policy specifications with very limited expressive power and are also not intuitive for use in mobile computing applications (for example the location abstractions they use are limited to IP addresses, or have insufficient granularity to cover relationships in a geographical reference space). Only the approach proposed in [10] discusses the idea of an extensible constraint enforcement mechanism that can potentially be used to satisfy the requirement of logical level access control outlined in Table 1. However, no specific extensions to support spatial constraint vocabulary have been provided.

We note that while the use of RBAC addresses the important requirement of scal-
ability to support our goal of simplified administration, the use of RBAC for location-based access control does pose other interesting challenges. For example, RBAC does not inherently address the issues of intuitiveness of location-based policy specification or enforcement of spatial constraints. We will address these requirements as part of the architectural design of Policy Mapper. In particular, we design a GUI to match the expressivity of GEO-RBAC using a visual representation to allow user-friendly construction of policies, and we provide a mechanism for management of user sessions in X-GTRBAC to handle the activation of roles when the associated constraints are dynamic in nature (such as continuously changing user position).

3 GEO-RBAC and X-GTRBAC

We now overview the main features of GEO-RBAC access control model and the X-GTRBAC policy specification language. Both GEO-RBAC and X-GTRBAC are based on the role-based access control (RBAC) standard [11].

3.1 GEO-RBAC

GEO-RBAC has been recently proposed as a comprehensive approach for modeling organizational roles in a mobile setting. GEO-RBAC extends RBAC with spatial access control capabilities that are much more flexible and expressive than those provided by related RBAC-based approaches [4, 5, 6]. In particular, GEO-RBAC is unique in its capability to account for multi-granularity of position and relationships in space, thereby providing an expressive spatial constraint vocabulary. These features make GEO-RBAC very well suited as the basis for designing the conceptual level of access control in Policy Mapper.

The core GEO-RBAC model is built upon three main concepts: spatial role, position model and role schema.

- **Spatial Role.** A spatial role is a spatially bounded organizational role. It is confined to a region called role extent which defines the boundaries of the spatial region. Users are then enabled to play the assigned roles only when located within the associated role extents. For example Nurse(CA237) is a role: Nurse is the role name and highway CA237 the role extent.

- **Position Model.** The position occupied by the user is described at two different levels called real and logical position respectively. The real position corresponds to the position of the user on Earth (such as longitude, latitude) acquired through some positioning technology, whereas the logical position not only has a geometric shape, but also a semantics (such as house, road). The logical position is obtained from a real position by applying an application-dependent function called location mapping function.

- **Role Schema.** A role schema defines some common properties of roles with a similar meaning, and a role instance is a spatial role defined over a specific extent, in compliance with the role schema. For example Nurse(CAHighway, SectorB, mSector)
is the schema for the nurse roles. In such schema, Nurse is the common name of a set of roles, whereas CAHighway denotes the type of the role extent; Sector denotes the type of logical position and $m_{Sector}$ denotes the location mapping function which computes the room in which the nurse is located based on the real position.

A sample fragment of policy describing role schemas and role instances for the preceding example is reported below. We use the following notation: $REXT_{FT}$ denotes the set of role extent types; $REXT$ denotes the set of role extents; $LPOS_{FT}$ denotes the set of logical position types; $m_*$ is the location mapping function. $R$, $R_S$, and $R_I$, denote respectively the set of role names, role schemas and role instances.

\[
R = \{\text{Doctor, Nurse, Pharmacist}\} \\
REXT_{FT} = \{\text{USHighway, CAHighway}\} \\
REXT = \{\text{US101, CA237}\} \\
LPOS_{FT} = \{\text{SectorA, SectorB}\}
\]

\[
R_S = \{ \text{Doctor(USHighway, SectorA, m_Sector),} \\
\quad \text{Nurse(CAHighway, SectorB, m_Sector),} \\
\quad \text{Pharmacist(CAHighway, SectorB, m_Sector)} \} \\
R_I = \{ \text{Doctor(US101),} \\
\quad \text{Nurse(CA237),} \\
\quad \text{Pharmacist(CA237)} \}
\]

### 3.2 X-GTRBAC

X-GTRBAC [2] is a well-known access control policy specification language with an associated enforcement architecture. X-GTRBAC was originally designed to capture the semantics of the RBAC model and its contextual extensions [2]. The language allows one to express a diverse set of contextual and logical constraints on roles.

Roles in X-GTRBAC have an articulated structure. Specifically, roles can be assigned a set of attributes through a credential, and a set of constraints. Depending on whether the constraints are satisfied or not, a role can be in an enabled or disabled state. A constraint in X-GTRBAC is defined as a logical expression using the usual $\lor$ and $\land$ operators on 3-tuples of the form $(y, \omega, \delta(p_1, ..., p_n))$ where $\delta$ is a parameterized function, $p_i$ and $y$ are either a role, permission, or a constant value, and $\omega = \{=, \neq, \geq, \leq, \in\}$. For example,

\[
(RoleLocation, =, hasCredAttrValue(RoleName, thisLocation))
\]

represents a logical expression that includes evaluation of a predicate function $hasCredAttrValue$ that compares the value of the $thisLocation$ attribute in credential of the role $RoleName$ using the $=$ operator with the expected value of $RoleLocation$. Using the XML-based syntax of X-GTRBAC, this constraint definition is represented as follows:

\[
<\text{LogicalExpression op="AND">}
\]
X-GTRBAC has a modular policy administration model and it can be customized to account for different types of contexts. In particular, the function definition in the constraint specification syntax allows functions for evaluation of multiple contexts to be plugged-in. This modularity has the implication that constraint specification mechanism of X-GTRBAC can be suitably extended to incorporate location-based constraints by providing the necessary logic for evaluation of spatial context. This feature will be used in Policy Mapper to provide the support for logical level of access control.

4 The Policy Mapper

We now describe the architecture of Policy Mapper. As shown in Figure 1, the architecture of Policy Mapper is based on two levels of access control, namely a conceptual level and a logical level. Using this architecture, we achieve the following tasks in support of our goal of simplified administration:

- Design conceptual level access control using GEO-RBAC
- Design an interface definition language for coupling conceptual level with logical level
- Provide evaluation logic in X-GTRBAC to accept the language

4.1 Conceptual Level Access Control Using GEO-RBAC

To design the conceptual level of access control, we restrict ourselves to consider the major components of a general GEO-RBAC policy, in particular the sets pertaining to roles and the assignment of permissions and users to roles.

Given a set $PRMS$ of permissions specified by the organization, and a set $U$ of users we represent the GEO-RBAC policy $pol_g$ as the tuple:

$$pol_g = <RS_g, R_g, PAS_g, PAI_g, UR_g>$$

where $RS_g$ is the set of role schemas, $R_g$ the set of role instances, $PAS_g$ and $PAI_g$ the permission-role_schema and permission-role_instance assignment relations respectively, $UR_g$ the user-role_instance assignment relation. We call this policy $G$-policy. We use the term $X$-policy to refer to a corresponding X-GTRBAC policy. Following RBAC standard, $X$-policy $pol_x$ is simply defined as the tuple

$$pol_x = <R_x, PA_x, UR_x>$$
where $R_x$ is the set of roles, $PA_x$ the set of permission-role assignment and $UR_y$ the user-role assignment relation.

Besides defining an intuitive specification, another task of the Policy Mapper is to offset the complexity introduced by the logical formalism needed to encode the expressive spatial constraints in the policy. Toward this end, we have developed a Graphical User Interface (GUI) that enables us to match the expressivity of GEO-RBAC using a visual representation to allow user-friendly construction of policies.

The GUI is implemented using the Java/SWING API. Essentially, the representation given to the administrator specifying the policy is a reference space containing a set of nodes representing mobile users, which we call clients, and another set representing locations spatially-relevant for access control, which we call targets. Clients are viewed as small rectangular regions, which can be moved about the map. Targets are equipped with appropriate location capture and verification technologies (such as RFID sensors). They have a defined location, as well as a range. Like the clients, they may be moved arbitrarily around the map. However, in a scenario like the one described in this paper, these will remain fixed points.

The GUI allows administrators to transpose the GEO-RBAC policy onto the reference space for visual representation and graphically composing the access control policy. The spatial constraints on roles are shown as geometric shapes on the GUI and are automatically translated to GEO-RBAC policy. A snapshot of the GUI is shown in Figure 2, together with the captions that demonstrate its use in the process of policy construction.

### 4.2 Formal Mapping and Interface Definition Language

The second task of Policy Mapper is to map the GEO-RBAC policy expressed at the conceptual level onto a language extension for X-GTRBAC that can be implemented at the logical level. This entails the definition of a formal mapping between the administrative commands of GEO-RBAC and X-GTRBAC. Such mapping is then used to design an interface definition language to implement the coupling between the conceptual and logical levels of access control.

The idea behind policy mapping is as follows. A G-policy is specified by the administrator at the conceptual level by issuing administrative commands (through the GUI or a command line) and is stored in a repository. The administrative commands are interpreted by the GEO-RBAC Admin Command Interpreter, which maps the command onto a sequence of X-GTRBAC administrative commands which modify the corresponding X-policy enforced at run-time. The latter task is performed by the X-GTRBAC Admin Command Interpreter. The Policy Mapper thus maps the GEO-RBAC administrative commands onto X-GTRBAC commands generating an equivalent X-policy. We say that the G-policy $p$ and the X-policy $p'$ are equivalent if the access requests which are satisfied in $p$ are all and only those which are satisfied in $p'$. We have defined a set of mapping rules by application of which we obtain an equivalent X-policy from a G-Policy. A notable mismatch between G-Policy and X-Policy is represented by the GEO-RBAC notion of role schema which is an extension to RBAC not supported by X-GTRBAC. To overcome this mismatch, we deal only with the semantics of the role schemas, which implies that a permission associated with a schema
is assigned to all instances of that schema. Therefore, role schemas do not exist in
the X-policy but the Policy Mapper maps the operations defined on role schemas onto
operations on sets of role instances corresponding to this role schema in the X-Policy.

The set of mapping rules collectively comprise the interface definition language that
implements the coupling between the conceptual and logical levels of access control.
We briefly comment the main operations and commands provided by the language, and
summarize them in Table 2.

- The `CreateRoleSchema` operation does not have any effect on the X-policy,
as role schemas are defined only in G-policy.

- The `CreateRoleInstance(r)` operation is interpreted as follows: since an in-
  stance `r` of schema `rs` inherits all the permissions which are assigned to `rs`, a
  new X-GTRBAC role `r'` is created and all permissions associated with `rs` are
  assigned to `r'`.

- A specular behavior characterizes the case in which a permission `p` is assigned to
  a schema `rs` through `AssignSPerm(rs, p)`: the permission `p` is then assigned
  to each X-GTRBAC role corresponding to an instance of `rs`.

- Command `AssignIPerm(rs, p)` simply maps a permission-to-role assign-
  ment in PAI onto a corresponding element in X-policy.

- Command `AssignRole(u, r)` assigns user `u` to the X-GTRBAC role corre-
  sponding to `r`.

- Command `AddActiveRole(s, r)` is a run-time command that allows a GEO-
  RBAC role `r` to be activated using an equivalent X-GTRBAC role in session `s`.

It should be noticed from Table 2 that the mapping of the GEO-RBAC role instance
onto the X-GTRBAC role is performed by the function `MapRole(r(e))`. This function
maps a role instance `r(e)`, of name `r` and extent `e`, into a X-GTRBAC role `rx` defined
as follows:

\[ rx = (<r, at, co>) \]

where `r` is the same name of the role, `at` the credential attribute representing the role
extent `e` and `co` the constraint stating the spatial containment between the user’s position
and the role extent specified in `att`.

4.3 Logic Level Access Control Using X-GTRBAC

The final task of Policy Mapper is to implement the conceptual level policy at the logi-
cal level using X-GTRBAC as shown in Figure 1. Toward this end, the constraint
processing capabilities of X-GTRBAC have been extended to allow processing of spa-
tial constraints encoded using GEO-RBAC. These extensions include specification and
evaluation of spatial context using functions that use geographical vocabulary. We call
this extended language the GEO-RBAC profile of X-GTRBAC.
The GEO-RBAC profile for X-GTRBAC relies on the use of the Geographical Markup Language (GML) [12] to represent the geographical data referred in the specification of spatial constraints. A notable extension to the X-GTRBAC processor required for processing GML data is the creation of data structures to store GML data and of the functions necessary to perform geometry calculations on these structures, and the extension of currently available parsers to take into account GML data. Additionally, extending the system while preserving constraint modularity and flexible administration requires late-binding of constraint processing routines and it has been achieved through the use of Java Reflection API.

The following code shows how the example GEO-RBAC policy is represented using X-GTRBAC. This constraint involves the evaluation of a spatial function `getRoadSector` with a return value being an attribute of GML type “Feature” which is used to represent a geographical area. X-GTRBAC recognizes this function and its parameters as belonging to the spatial domain, and processes them according to the semantics of the GEO-RBAC profile of X-GTRBAC. This constraint evaluates to true if the user submitting the request is included in an area of highway defined as `SectorA` as per the GML definition.

```xml
<LogicalExpression op=" AND">
  <Predicate>
    <Operator>contained_in</Operator>
    <FuncName>getHighwaySector</FuncName>
    <RetValue>
      <Attribute name="SectorA"/>
      <Feature>
        <gml:name>SectorA</gml:name>
        <gml:description>Highway Sector defined as SectorA</gml:description>
        <gml:extentOf>
          <gml:Envelope>
            <gml:lowerCorner>0 0</gml:lowerCorner>
            <gml:upperCorner>100 100</gml:upperCorner>
          </gml:Envelope>
        </gml:extentOf>
      </Feature>
    </RetValue>
  </Operator>
</LogicalExpression>
```

Table 2: Commands and operations for the interface definition language

<table>
<thead>
<tr>
<th>Commands and Operations</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateRoleSchema&lt;rs&gt;, rs ∈ RS</td>
<td>NULL.</td>
</tr>
<tr>
<td>CreateRoleInstance&lt;rs, r&gt;, r ∈ RI</td>
<td>CreateRole&lt;rs(MapRole(r)) For all p ∈ S_PrmsAssignment&lt;rs(SchemaOf(r)) do: AssignPerm&lt;rs(MapPerm(r,p))</td>
</tr>
<tr>
<td>Assign_SPerm&lt;rs,p&gt;, rs ∈ RS, p ∈ PRMS</td>
<td>For all r ∈ RI, SchemaOf(r) = rs do: AssignPerm&lt;rs(MapPerm(r,p))</td>
</tr>
<tr>
<td>Assign_IPerm&lt;r,p&gt;, r ∈ RI, p ∈ PRMS</td>
<td>AssignPerm&lt;r(MapPerm(r,p))</td>
</tr>
<tr>
<td>AddActiveRole&lt;s,r&gt;, r ∈ RI, s ∈ S</td>
<td>AddActiveRole&lt;s,MapRole(r))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commands and Operations</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateRole&lt;rs, r&gt;</td>
<td>CreateRole&lt;rs(MapRole(r)) For all p ∈ S_PrmsAssignment&lt;rs(SchemaOf(r)) do: AssignPerm&lt;rs(MapPerm(r,p))</td>
</tr>
<tr>
<td>Assign_SPerm&lt;r,p&gt;, r ∈ RI, p ∈ PRMS</td>
<td>For all r ∈ RI, SchemaOf(r) = rs do: AssignPerm&lt;rs(MapPerm(r,p))</td>
</tr>
<tr>
<td>Assign_IPerm&lt;r,p&gt;, r ∈ RI, p ∈ PRMS</td>
<td>AssignPerm&lt;r(MapPerm(r,p))</td>
</tr>
<tr>
<td>AddActiveRole&lt;s,r&gt;, r ∈ RI, s ∈ S</td>
<td>AddActiveRole&lt;s,MapRole(r))</td>
</tr>
</tbody>
</table>
Once the policy is properly processed and interpreted by X-GTRBAC, the system can then process run-time access requests by users. Effectively, the system needs to handle run-time commands issued on the GEO-RBAC policy by mapping them to appropriate X-GTRBAC run-time commands using the interface definition language. One example of a run-time command is the last command given in Table 2, which allows a role to be activated by a user in a given session. A session database SessionDB shown in Figure 1 maintains status information on the current sessions.

The mobility of users raises the issue of handling activation and deactivation of roles in a session. For this purpose, X-GTRBAC provides facilities for session management, which allow the deactivation of a role to be automatically triggered when the location constraint is no longer satisfied. With reference to the GUI, the location capture technology at the target is queried periodically by the system to update the location information of the client. As soon as the client moves out of the range of the target, any roles activated by the user based on proximity to this target are automatically deactivated.

5 Concluding remarks

The development of Policy Mapper allows us to bridge a critical gap with regards to expressiveness and enforcement of spatial constraints in location-based access control policies. It allows us to define access control at a conceptual level and a logical level to carry out the tasks of constraint specification and enforcement, respectively, and also provides an interface definition language to allow the coupling of the two levels. Future work includes detailed evaluation of the system using real world location-based access control policies in mobile computing applications.

References


Figure 2: Snapshots of GUI for specification of GEO-RBAC policy (from top to bottom, left to right): (a) mapping the spatial geometry corresponding to the policy, (b) adding the targets on the map, (c) mapping client sessions on the GUI (d) mapping movement of clients on the GUI to simulate mobility.