Enforcing Protection Mechanisms for Geographic Data

Alban Gabillon, Patrick Capolsini

Enforcing Protection Mechanisms for Geographic Data

Alban Gabillon and Patrick Capolsini

Université de la Polynésie Française, BP 6570, 98702 FAA'A, French Polynesia
{Alban.Gabillon,Patrick.Capolsini}@upf.pf

Abstract. In the framework of a geographic application displaying maps, there are several solutions for protecting a sensitive object. Sensitive objects can be hidden, masked, blurred or even replaced by fake objects. In this paper we suggest a framework to specify protection mechanisms to enforce whenever a prohibition is derived from the security policy. This framework includes (i) logical rules allowing us to derive protection mechanisms from prohibitions, and (ii) an algorithm which builds the map to display, according to the derived protection mechanisms.

Keywords: Access Control, Geo-spatial Data visualization, Map service, Policy Enforcement Point.

1 Introduction

Given a query, the security policy of a database application specifies which objects are authorized and which objects are unauthorized. In a traditional database approach, enforcement of the security policy is simply done by removing the unauthorized objects from the final answer to the query. In a geographic database application, things are more complicated. Let us consider a map service building maps from various spatial objects. In such an application, there are several methods for protecting unauthorized objects. Some unauthorized objects are simply removed from the final map as it is the case in a traditional database application, but some other sensitive objects are protected by using methods which are specific to geographic applications. Examples of such specific methods are referred to as *blurring*, *masking*, *pixelization* or "*cut and paste*" (i.e. overlay a sensitive object with another fake objects). Figure 1 shows some examples of sensitive objects which were protected by using such methods in well known geographic applications. These examples are all taken from [1]. The top left map shows a masked area at the state border between Yukon and Alaska. The top right map shows a pixelizated factory at Toulouse in southwestern France. On the bottom left map, part of the Michael Army Airfield (Utah) is blurred. On the bottom right map a fake landscape object overlays a sensitive military area of Xinshe in Taiwan.

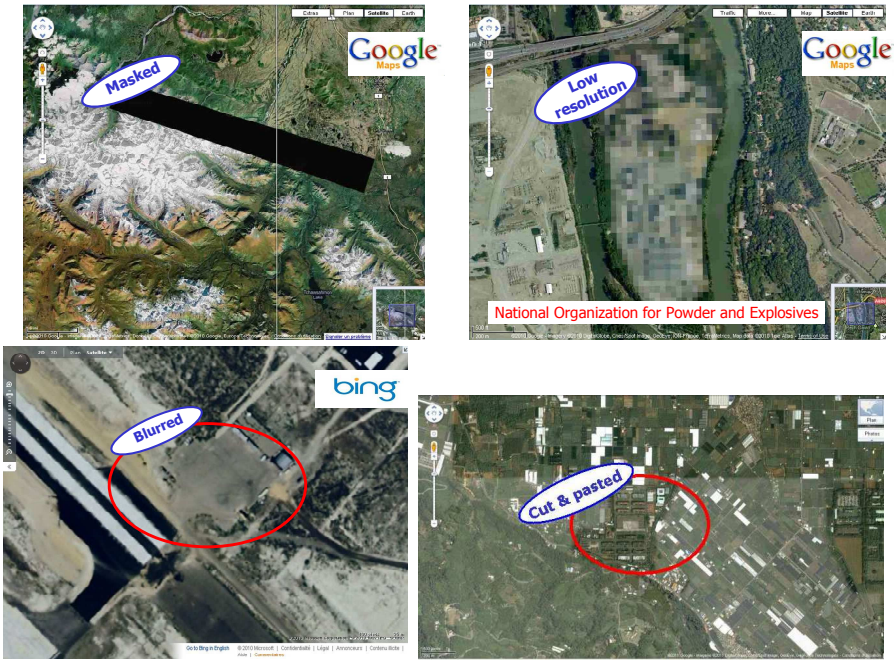

Fig. 1. Sensitive spatial objects protected with various methods

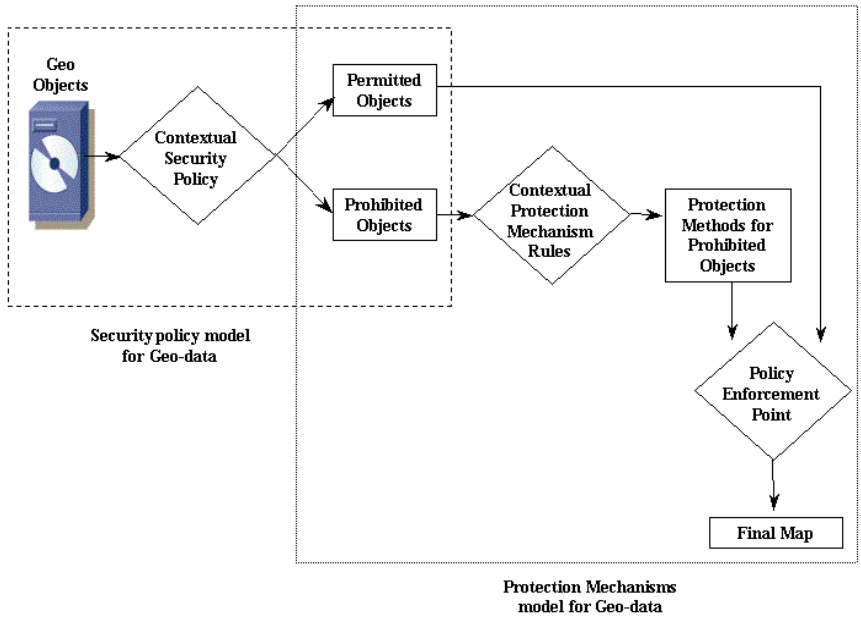

Fig. 2. Protection Mechanisms Model

Existing security models [2-5] for geo-spatial applications focus on how to express security policies. Figure 2 shows that the work presented in this paper is located downstream of these models. The studies cited above deal with what is represented inside the dashed rectangle. They provide models for expressing contextual security policies for geographic data. Given a query that addresses a set of spatial objects, the security policy determines which objects are authorized and which objects are not authorized. The work presented in this paper deals with what is inside the dotted rectangle. It breaks down as follows:

- It defines a formal framework for deriving security mechanisms to be enforced on unauthorized objects.
- It proposes a Policy Enforcement Point (PEP) algorithm to display the final map taking into account the protection mechanisms that should be enforced on unauthorized objects.

To our knowledge, we are the first to propose a complete formal framework for specifying the mechanisms that should be enforced on prohibited objects. Let us mention, that we published a preliminary version of this work as a position paper in [6].

In section 2 of this paper, we felt the need to recall the basics of a security model we already defined in [4] and [5]. This security model allows us to express contextual security policies for geographic applications. In section 3 we define our logical framework for specifying *protection rules*. In section 4, we define the PEP algorithm which enforces *protection mechanisms* and builds the map to display. In section 5 we illustrate our proposal with a complete application example. In section 6, we give a sketch of the implementation of our proposal within the framework of the OpenGIS® Styled Layer Descriptor (SLD) Profile [7] of the OpenGIS® Web Map Service (WMS) Encoding Standard [8]. A prototype showing the feasibility of our approach can be found at the following url:
http://pages.upf.pf/Patrick.Capolsini/rech/protect/index.htm.
In section 7, we review related works. Finally section 8 concludes this paper.

2 Security Policy Model

In [4] and [5], we proposed a security policy model for geographic applications, based on the OrBAC model [9] and the ABAC model [10]. Our model considers dynamic spatial security rules. A spatial dynamic security rule can be activated or deactivated depending on some *spatial context*. Generally, a spatial context is considered to be a spatial condition that holds on the subject and/or the object. In [4], we identified and modelled various types of spatial contexts based on the user location and/or the spatial object location. We also showed how to model geo-temporal contexts and contexts related to movement. In [5], we focused on visualization of geo-data i.e. we showed how to model various types of *visualization contexts* (such as zoom-in factor, layers transparency, brightness …etc) for geo-data and how to express dynamic security rules based on such contexts.

Since this paper focuses on policy implementation and not on the security policy itself, we present here a simplified version of our model. It mainly defines a logical

language for expressing security policies for geographic applications. Note, however, that we also use the predicates and functions defined in this section to express our protection rules in section 3.
In section 2.1 we define the objects of our model. In sections 2.2 and 2.3 we define elements of our language for writing security policies. In section 2.4, we define the concept of spatial query. In section 2.5 we define authorization rules.

2.1 Geometric Entities

A *georeferenced (geometric) object* is a granule of information that is relevant to an identifiable subset of the Earth's surface [11]. Any geometric object has the following two components [12] : a *description*: the entity is described by a set of descriptive attributes (e.g. the name of a city) and a *geometry* which indicates the entity's location and its shape. The geometry model we consider is the OpenGIS Geometry Model [13].

2.2 Spatial Analysis Functions

Spatial analysis functions take one or more geometric objects as input and return either a number or another geometric object. We consider the following functions. Let *a* and *b* be two geometric objects and *x* a scalar:

- *distance(a,b)* – Returns the shortest distance (a scalar) between any two points in the two geometric objects *a* and *b*
- *buffer(a,x)* – Returns a geometric object that represents all points whose distance from geometric object *a* is less then or equal to *x*
- *convexHull(a)* – Returns a geometric object that represents the convex hull (mathematical definition) of geometric object *a*
- $a \cap b$, $a \cup b$, $a \setminus b$, $a \Delta b$, – Respectively returns a geometric object that represents the point set intersection (resp. union, resp. difference, resp. symmetric difference) of object *a* with object *b*
- *I(a)*, *B(a)*, *E(a)* and *dim(a)* respectively returns the interior, boundary, exterior and dimension (-1 for the empty geometry Ø, 0 for *Point*, 1 for *Linestring* and 2 for *Polygon*) of *a*.
- *speed(a)* – Returns the speed of the object. The speed is a scalar value greater than or equal to 0.
- *direction(a)* – Returns the direction taken by the object. The direction is an angle value between 0 and 360 degrees. It is equal to N/A (Not Applicable) if the speed is equal to 0.

2.3 Spatial Predicates

Spatial predicates are used to test for the existence of a specified topological relationship between two geometric entities. Using functions *I(g)* and *dim(g)* returning respectively the interior and dimension of geographic object *g*, [13] defines eight spatial predicates namely, *Equals*, *Disjoint*, *Intersects*, *Touches*, *Crosses*, *Within*, *Contains* and *Overlaps*

$$\forall g_1, \forall g_2, Equals(g_1, g_2) \leftrightarrow (g_1 \cap g_2) = (g_1 \cup g_2) \quad (1)$$

$$\forall g_1, \forall g_2, Disjoints(g_1, g_2) \leftrightarrow g_1 \cap g_2 = \varnothing \quad (2)$$

$$\forall g_1, \forall g_2, Touches(g_1, g_2) \leftrightarrow (I(g_1) \cap I(g_2) = \varnothing) \wedge (g_1 \cap g_2 \neq \varnothing) \quad (3)$$

$$\forall g_1, \forall g_2, Crosses(g_1, g_2) \leftrightarrow \quad (4)$$
$$(dim(I(g_1) \cap I(g_2)) < max(dim(g_1), dim(g_2))) \wedge (g_1 \cap g_2 \neq g_1) \wedge (g_1 \cap g_2 \neq g_2)$$

$$\forall g_1, \forall g_2, Within(g_1, g_2) \leftrightarrow (g_1 \cap g_2 = g_1) \wedge (I(g_1) \cap E(g_2) \neq \varnothing) \quad (5)$$

$$\forall g_1, \forall g_2, Contains(g_1, g_2) \leftrightarrow Within(g_2, g_1) \quad (6)$$

$$\forall g_1, \forall g_2, Overlaps(g_1, g_2) \leftrightarrow (dim(I(g_1)) = dim(I(g_2)) = dim(I(g_1) \cap I(g_2))) \quad (7)$$
$$\wedge (g_1 \cap g_2 \neq g_1) \wedge (g_1 \cap g_2 \neq g_2)$$

$$\forall g_1, \forall g_2, Intersects(g_1, g_2) \leftrightarrow \neg Disjoint(g_2, g_1) \quad (8)$$

2.4 Spatial Query

In the framework of a map service a spatial query outputs a map. This map is constructed from a set of geo-referenced objects which are all displayed at the same zoom-in factor. This zoom-in factor is a parameter of the query.
Let *q* be a spatial query. We denote *O(q)*, the set of objects addressed by query *q* and *zf(o)* the zoom-in factor of object *o*. This zoom-in factor is inherited from query *q* and is the same for all objects addressed by query *q*.

2.5 Contextual Authorization Rules

Security rules specify how subjects can execute actions on objects. Our model includes permissions (positive rules) and prohibitions (negative rules). Given a query, authorized objects addressed by the query are used to build up the map.
We define a positive authorization rule as a logical rule having the following form:

$$\forall s \forall o (Condition \rightarrow Permit(s, o)) \quad (9)$$

Permit(s,o) reads "s is permitted to view object o."
We define a negative authorization rule as a logical rule having the following form:

$$\forall s \forall o (Condition \rightarrow Deny(s, o)) \quad (10)$$

Deny(s, o) reads "s is forbidden to view object o."
In both rules *Condition* is a logical expression used to express some properties regarding the subject, the object and the context.

Let us consider the following example of security policy which consists of the four following rules:
The first security rule says that civilians are forbidden to view tanks.

$$\forall s \forall o \big(Civilian(s) \wedge Tank(o) \rightarrow Deny(s,o) \big) \tag{11}$$

The second security rule says that civilians are forbidden to view barracks at a zoom-in factor greater than 1.

$$\forall s \forall o \big(Civilian(s) \wedge Barrack(o) \wedge zf(o) > 1 \rightarrow Deny(s,o) \big) \tag{12}$$

The third security rule says that soldiers have the permission to view tanks:

$$\forall s \forall o \big(Soldier(s) \wedge Tank(o) \rightarrow Permit(s,o) \big) \tag{13}$$

The fourth security rule says that soldiers do not have the permission to view tanks which are not within the military zone:

$$\forall s \forall o \big(Soldier(s) \wedge Tank(o) \wedge \neg Within(o, MilitaryZone) \rightarrow Deny(s,o) \big) \tag{14}$$

Note that there is a conflict between the last two rules regarding tanks which are not within *MilitaryZone* area. It is not the purpose of this paper to discuss this issue. The reader can refer to [4], where we use the conflict resolution strategy defined in the OrBAC model. This conflict resolution strategy is based on separation constraints and priorities assigned to rules. Our first aim, in this paper, is to devise a logical framework to specify the security mechanisms which are to be enforced whenever we derive an instance of the *Deny* predicate from the security policy, regardless of the conflict resolution strategy which is used. We define this framework in the next section.

3 Contextual Protection Rules

3.1 Definition

In this section, we define a complete framework for specifying the protection mechanism which should be enforced in case a user is denied to view a given object. The logical language we use is based on the language we defined in the previous section.
A *Protection Rule* is a rule of the form:

$$\forall s \forall o \big(Condition \wedge Deny(s,o) \rightarrow Protect(o,M) \big) \tag{15}$$

As it is the case with authorization rules, *Condition* is used to express some properties that should hold on the subject, the object and the context. This means in particular that given an unauthorized object, the protection mechanism that should be enforced may vary from one context to another.

Protect(o,M) reads "*o* should be protected with mechanism *M*". *M* is a protection mechanism function which is one of the followings:
Let *g* be a geometric object and *i* a scalar:

- *reject_query*: reads "reject the query which requires *o* to be displayed" i.e. empty map is returned even if the query addressed some authorized objects.
- *blur* : reads "blur *o*"
- *mask*: reads "mask *o*"
- *pixelizate*: reads "lower *o*'s resolution".
- *hide*: reads "remove *o*"
- *paste(g)*: reads "cut and paste g" i.e. "overlay *o* with *g*". . *paste(g)* is very often used to build what is referred to as a *cover story* i.e. a lie. It can be a fake object which does not exists in the real world. It can be an existing object from which some visual details have been hidden. It can be an existing object, but shown at an incorrect location etc.
- *zoom_in(i)*: reads "forces the zoom-in factor of object *o* to a value which is less than or equal to *i*". As we will see in section 4, this protection method would also decrease the zoom-in factor of other objects, even authorized ones, since given a map the zoom-in factor is the same for all objects.

Note that, to define our model, we do not need to enter into the details of the *blur*, *mask* and *pixelizate* functions. However, in a real implementation, these protection mechanisms would require some parameters such as the intensity for the *blur* function, the shape and the position of the mask for the *mask* function and the resolution for the *pixelizate* function.

Let us consider the following two examples of protection rules:

$$\forall s \forall o \left(Tank\ (o) \wedge Deny\ (s,o) \rightarrow Protect\ (o, hide) \right) \tag{16}$$

$$\forall s \forall o \left(Barrack\ (o) \wedge Deny\ (s,o) \rightarrow Protect\ (o, zoom\_in(1)) \right) \tag{17}$$

The first rule says that if someone who is forbidden to see tanks request to see them then tanks should be removed from the returned map. The second rule says that someone who is forbidden to see barracks can in fact see them but at a zoom-in factor equal to 1.
If for a given prohibition there is no specific protection rule then a default mechanism applies. This default mechanism depends on the application. For example the following default rule says that the default mechanism is *blur*.

$$\forall s \forall o \left(Deny\ (s,o) \rightarrow Protect\ (o, blur) \right) \tag{18}$$

If for a given prohibition, several mechanisms can be derived then only one of them should be selected. Such selection should be done on a priority basis. However, we have two options for assigning priorities:

- either we assign priorities to the mechanism themselves. For example, the following list could represent the hierarchy of mechanisms (from the lowest priority to the highest priority): {*zoom-in, pixelizate, blur, mask, paste, hide, reject_query*},

- or we assign priorities to protection rules (with the default rule having the lowest priority).

In section 4, we design a Policy Enforcement Point (PEP) algorithm which works with both approaches.

3.2 Merging Prohibitions and Protection Rules

In our model, we distinguish between the prohibitions and the protection rules. However, we could envisage merging the two types of rules as follows:

$$\forall s \forall o \left(Condition \rightarrow Deny(s,o) \wedge Protect(o,M) \right) \tag{19}$$

In the above definition, we directly specify in the prohibition rule the protection mechanism that should be enforced. If we apply this principle to the example described in section 3.1, then rules 11, 12 and 14 are merged with rules 16 and 17 as follows:

$$\forall s \forall o \left(Civilian(s) \wedge Tank(o) \rightarrow Deny(s,o) \wedge Protect(o,hide) \right) \tag{20}$$

$$\forall s \forall o \left(\begin{array}{l} Civilian(s) \wedge Barrack(o) \wedge zf(o) > 1 \\ \qquad \rightarrow Deny(s,o) \wedge Protect(o, zoom\_in(1)) \end{array} \right) \tag{21}$$

$$\forall s \forall o \left(\begin{array}{l} Soldier(s) \wedge Tank(o) \wedge \neg Within(o, MilitaryZone) \\ \qquad \rightarrow Deny(s,o) \wedge Protect(o,hide) \end{array} \right) \tag{22}$$

The obvious advantage of merging prohibitions and protection rules is that we end up with managing only one set of rules. However, this approach has the following disadvantages:

- If we need to enforce the same protection mechanism for several different prohibitions then we have to specify this mechanism in each of the prohibition rules. For example, we had to specify that the protection mechanism should be *hide* in rules 20 and 22.
- We reduce the expressive power of our model. In rule 19, the same condition triggers both the prohibition and the protection mechanism, whereas in rules 10 and 15, the condition triggering the prohibition can be different from the condition triggering the protection mechanism.

In the remainder of this paper we will not consider any more the possibility of merging the two types of rules since we want our model to have the highest possible expressive power. However, from a practical point of view, we are perfectly aware that a single set of rules might be easier to manage than two separate sets of rules.

4 Policy Enforcement Point Algorithm

In this section we define an algorithm for (i) enforcing protection mechanisms and (ii) construct the map to display. *O(q)* denotes the set of objects addressed by query *q*. *zf(q)* denotes the zoom-in factor of query *q* (see section 2.4). *map* denotes the map to construct. *empty_map* denotes the empty map. *minzf* denotes the zoom-in factor at which the final map is going to be displayed. *insert(o,map)* denotes a procedure which inserts geo-referenced object *o* into map *map*. *overlay(o,g)* denotes a procedure which overlays geo-referenced object *o* with geo-referenced object *g*. *mask(o)* denotes a function which overlays *o* with a mask, *blur(o)* denotes a function which blurs *o*, *pixelizate(o)* denotes a function which lowers *o*'s resolution. *applyzf(i,map)* is a function which applies zoom-in factor *i* on map *map*

```
/* Protect(o,M) should be read "Protect(o,M) can be derived from
the Protection Rules" */
1.  map ← empty_map
2.  minzf ← zf(q)
3.  For o in O(q)
4.     If Protect(o,reject_query) then
5.        Return(empty_map)
6.     Else
7.        If NOT Protect(o,hide) then
8.           insert(o,map)
9.           If  Protect(o,paste(g)) then
10.             overlay(o,g)
11.          Else
12.             If Protect(o,mask) then
13.                mask(o)
14.             Else
15.                If Protect(o,blur) then
16.                   blur(o)
17.                Else
18.                   If Protect(o,pixelizate) then
19.                      pixelizate(o)
20.                   Else
21.                      If Protect(o,zoom_in(i)) then
22.                         minzf ←min(i,minzf)
23. Return(applyzf(minzf,map))
```

This algorithm works in the following two cases:

- Mechanisms have different priorities and the following mechanism hierarchy is used (from the lowest priority to the highest priority): *{zoom-in, pixelizate, blur, mask, paste, hide, reject_query}*. The algorithm is designed to select the highest priority mechanism in case more than one mechanism can be derived from a single prohibition.
- Priorities are assigned to protection rules (with the default rule having the lowest priority). The algorithm selects the mechanism derived from the highest priority rule in case more than one mechanism can be derived from a single prohibition. However, it might happen that several mechanisms are selected. This can be the case if some protection rules have the same priority. If this occurs, then the

algorithm selects the mechanism to enforce on the basis of the mechanisms hierarchy.

Regarding this algorithm we can make the following comments:

- If, for any object, mechanism, *reject_query* should be enforced then the algorithm terminates (line 5) and an empty map is returned, even if the query addressed some authorized objects.
- Prohibited objects that should be removed from the final map are ignored (line 7).
- Line 8 inserts authorized objects. It also inserts prohibited objects on which a protection mechanism should be applied.
- The returned map is displayed at the lowest zoom-in factor imposed by protection rules (line 22). For example, let the zoom-in-factor of the query be equal to 5. Assume there are two objects addressed by the query which are protected and should be displayed respectively at zoom-in factor equal to 4 and zoom-in factor equal to 3. The lowest zoom-in factor imposed by protection rules is selected and the map is displayed at zoom-in factor equal to 3.
- Lines 10, 13, 16 and 19 apply various protection methods.
- Line 23 applies the zoom-in factor and returns the final map.
- This algorithm is linear with the number of objects addressed by the query.

Of course this algorithm could be written differently. We could consider another mechanism hierarchy or we could consider a partial order on the set of mechanisms. In this latter case, if two mechanisms which cannot be compared can be derived from the same prohibition then priorities on rules should be used to select one of these mechanisms.

5 Application Example

5.1 Contextual Security Policy

We consider an organization simultaneously managing a fleet of taxis and a fleet of ambulances. While driving, drivers from this company use a spatial application displaying surrounding objects. Fig 3 shows that subjects are drivers who can be either taxi drivers or ambulance drivers. Objects are buildings, roads and military areas (including military hospitals). Basically, the security policy expresses the fact that drivers can view spatial data which are within a radius of 40 km around their position. However, there are some restrictions to this general rule.

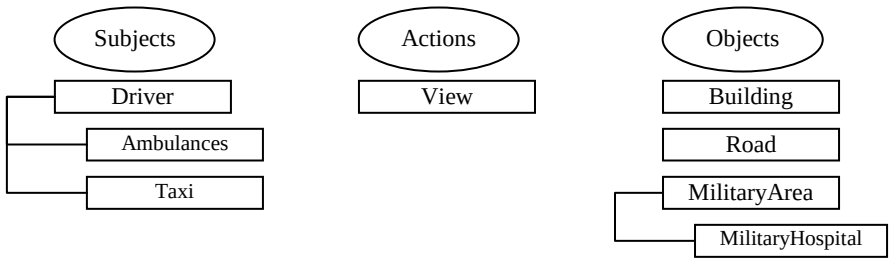

Fig. 3. Synopsis of our example

Default policy: The default policy is closed i.e. given a subject *s* and an object *o*, if *Permit(s,o)* cannot be derived from the security policy then *Deny(s,o)* should be derived.
Drivers have the permission to view at a maximum zoom-in factor of 10 any object that is within a radius of 40km around their position.

$$\forall s \forall o \left(Driver(s) \wedge distance(s,o) \leq 40 \wedge zf(o) \leq 10 \rightarrow Permit(s,o) \right) \quad (23)$$

Drivers have the permission to view roads at a maximum zoom-in factor of 10 (even those which are not within a radius of 40km).

$$\forall s \forall o \left(Driver(s) \wedge Road(o) \wedge zf(o) \leq 10 \rightarrow Permit(s,o) \right) \quad (24)$$

Taxis driving at a speed greater than 100km per hour are forbidden to view any object. This rule does not apply to ambulances since they are emergency vehicles.

$$\forall s \forall o \left(Taxi(s) \wedge speed(s) \geq 100 \rightarrow Deny(s,o) \right) \quad (25)$$

Drivers are prohibited to view military areas .

$$\forall s \forall o \left(Driver(s) \wedge MilitaryArea(o) \rightarrow Deny(s,o) \right) \quad (26)$$

Drivers are prohibited to view buildings which are contiguous to military areas

$$\forall s \forall o \forall m \left(Driver(s) \wedge Building(o) \wedge MilitaryArea(m) \wedge Touches(o,m) \rightarrow Deny(s,o) \right) \quad (27)$$

Ambulances are permitted to view military hospitals at a maximum zoom-in factor of 5.

$$\forall s \forall o \left(Ambulance(s) \wedge MilitaryHospital(o) \wedge zf(o) \leq 5 \rightarrow Permit(s,o) \right) \quad (28)$$

The above security policy may lead to conflicts. Rule 23 and rule 24 conflict with the default policy. For a taxi driving at more than 100 km per hour, rule 25 conflicts with rules 23 and 24. For military areas, rule 26 conflicts with rule 23. For buildings contiguous to military areas, rule 27 conflicts with rule 23. For ambulances, military hospitals and a zoom-in factor lower than 5, rule 28 conflicts with rule 26. As we said before, it is not the purpose of this paper to discuss conflict resolution. In this example, we simply assume that rules 23 and 24 override the default policy, rule 25 overrides rules 23 and 24, rule 26 overrides rule 23, rule 27 overrides rule 23 and rule 28 overrides rule 26.

5.2 Protection Rules

Default mechanism: We define the default mechanism as *hide*:

$$\forall s \forall o \left(Deny(s,o) \rightarrow Protect(o, hide) \right) \quad (29)$$

Rule 30 says that if a taxi driving at 100km is forbidden to view an object then his query should be rejected

$$\forall s \forall o \left(Taxi(s) \wedge speed(s) \geq 100 \wedge Deny(s,o) \rightarrow Protect(o, reject\_query) \right) \quad (30)$$

Rule 31 says that if a subject is forbidden to view a building within a radius of 40km then the resolution of this building should be lowered.

$$\forall s \forall o \left(Building(o) \wedge distance(s,o) \leq 40 \wedge Deny(s,o) \rightarrow Protect(o, pixelizate) \right) \quad (31)$$

Rule 32 says that if a subject is forbidden to view a military area within a radius of 40km then this military area should be masked.

$$\forall s \forall o \left(MilitaryArea(o) \wedge distance(s,o) \leq 40 \wedge Deny(s,o) \rightarrow Protect(o, mask) \right) \quad (32)$$

Rule 33 says that if an ambulance is forbidden to view a military hospital within a radius of 40km then the zoom-in factor should be lowered to 5.

$$\forall s \forall o \left(\begin{array}{c} MilitaryHospital(o) \wedge Ambulance(s) \wedge distance(s,o) \leq 40 \wedge Deny(s,o) \\ \rightarrow \text{Protect}(o, zoom\_in(5)) \end{array} \right) \quad (33)$$

We assign priorities to protection rules. The default rule 29 has the lowest priority. Rule 30 has the highest priority. Rules 31 and 32 have the same priority. Rule 33 has a higher priority than rule 32.

The default mechanism applies whenever it is not possible to derive any mechanism for a given prohibition. Therefore we can easily see that the default mechanism (rule 29) applies to instances of the *Deny* predicate which are derived from the default (closed) policy. These instances address objects which are not the roads and which are outside of a 40km radius. Rule 30 applies to instances of the *Deny* predicate which are derived from rule 25. Taxis driving too fast should see their query rejected i.e. taxis are in fact forbidden to use the application as long as they drive too fast. Rule 31 applies to the instances of the *Deny* predicate which are derived from rule 27, i.e. buildings touching military areas should be pixelizated. Rule 33 applies to *some* instances of the *Deny* predicate which are derived from rule 26. These instances address ambulances requesting to view military hospital at a zoom-in factor greater than 5 (recall that rule 28 say that ambulances are permitted to view military hospitals at a zoom-in factor lower than 5). Rule 32 applies to all the other instances of the *Deny* predicate which are derived from rule 26.

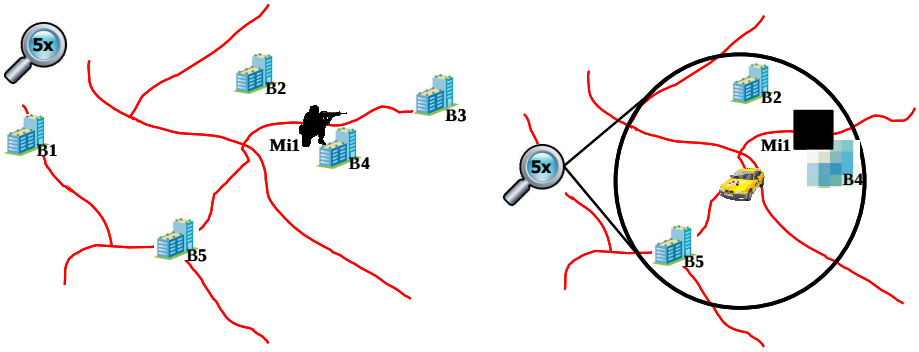

Fig. 4. Original map and Taxi driver view

The left picture of figure 4 shows an example of an original road map. The right picture shows the taxi driver view of the same map. The zoom-in factor is 5, the circle represents the 40km radius, objects outside this radius are hidden (except roads), the military area is masked and the building near the military area is pixelizated. Note that

under the assumption that the military area is not a military hospital, the ambulance view is the same as the taxi view.

6 Sketch of Implementation

In this section, we sketch the implementation of our model within the framework of the OpenGIS® Styled Layer Descriptor (SLD) [7] Profile of the OpenGIS® Web Map Service (WMS) [8] specification.

6.1 The OpenGIS® Web Map Service specification

Among all the specifications published by the OGC [14] regarding Geographic Databases and data exchange protocols, the most popular and widely used is undoubtedly the OpenGIS® Web Map Service (WMS). WMS servers support the creation and display of registered and superimposed map-like views of information coming from multiple heterogeneous sources including other WMS servers. The underlying protocol for WMS is the Hypertext Transfer Protocol (HTTP). Most WMS servers implement a common gateway interface (cgi-bin) and may be requested via an URL issued from a standard web-browser or any WMS-enabled software. The main parameters of a basic *GetMap* request to a WMS server include an ordered (bottom to top) list of layers (spatial objects), an ordered list of styles in which each layer is to be rendered (with a one-to-one correspondence with the list of layers), a Bounding Box specifying the geographical extent of the region to map and two parameters (*width* and *height*) specifying the final size of the requested image. The response of the server to a valid WMS *GetMap* request is an image file in the specified format (MIME type such as PNG, GIF or JPEG) having the dimension *width* by *height* pixels. Two points are fundamental in WMS requests: (i) conjunction of the requested image size (*width* and *height* parameters) with the *in situ* geographical extent (Bounding Box parameter) leads to the definition of the zoom-in factor for the final map and (ii) applying a specific style to a specific layer (geographical object) leads to the concept of Styled Layers developed in the next subsection.

6.2 The OpenGIS® Styled Layer Descriptor profile

A styled layer represents a particular combination of a ‘layer’ and a ‘style’ in which that layer can be symbolized. Conceptually, the layer defines a stream of features and the style defines how those features are symbolized. Defined by OGC in 2007, the Styled Layer Descriptor (SLD) is an XML-based description format for formatting data from a WMS flow. It plays the same role as a CSS file to an HTML page, the goal being to completely separate the style from the data. For example the same geographic object of type point may be symbolized as a small blue dot, a large red cross or a medium green square. A polygonal object may be drawn as a light blue transparent hatch, a fully opaque black polygon as well as a green grass-looking textured object. Named-styles are predefined using SLD files and used within the *Styles* parameter of the WMS request.

6.3 Rewriting WMS queries

Our prototype acts as a front-end engine rewriting WMS queries issued from an authenticated user on the basis of the outcome produced by the PEP algorithm presented in section 4. User queries are rewritten as follows:

- Each object that should be hidden (line 7 of the PEP algorithm) is simply removed from the list of requested layers.
- Each object *o* that should be overlaid by another object *g* (line 11 of the algorithm) is replaced by object *g* in the list of requested layers.
- Reducing the zoom-in factor (line 23 of the algorithm) is achieved by modifying the width and height of the final image so that the ratio between the bounding box and the size of the image respects the zoom-in factor imposed by the PEP algorithm.
- We wrote three SLD *protection styles* simulating respectively the three protection mechanisms *blur*, *mask* and *pixelizate* (line 14, 17 and 20 of the PEP algorithm). For each object that should be blurred, masked or pixelizated, the corresponding protection style replaces the style of the original query.

Let us assume for example that the following query is issued by a taxi driver. It requests all layers (objects) with the default style for each layer (see Figure 4).

```
http://yourWmsServer.com/wms?SERVICE=wms&VERSION=1.3.0&REQUEST=G
etMap&LAYERS=Roads,B1,B2,B3,B4,B5,Mi1&STYLES=&BBOX=x1,y1,x2,y2&W
IDTH=600&HEIGHT=600&FORMAT=image/png&SRS=epsg:4326
```

This query would be rewritten as follows by our front-end engine:

```
http://yourWmsServer.com/wms?SERVICE=wms&VERSION=1.3.0&REQUEST=G
etMap&LAYERS=Roads,B2,B4,B5,Mi1&STYLES=,,PixelSLD,,MaskSLD&BBOX=
x1,y1,x2,y2&WIDTH=600&HEIGHT=600&FORMAT=image/png&SRS=epsg:4326
```

The rewritten query addresses buildings within a radius of 40km, requests the military area Mi1 with the mask style *MaskSLD*, requests the building *B4* with the pixelizate style *PixelSLD* and requests other objects with the server default style.
Currently, our prototype (http://pages.upf.pf/Patrick.Capolsini/rech/protect/index.htm) is only at the proof-of-concept stage. It implements the *mask* and *pizelizate* SLD and uses some predefined examples. Our prototype considers a set of prohibited objects. First, it asks the user to set some context parameters. Second it shows how the original user WMS query is rewritten into a secure WMS query. Third, it displays the map returned by the map engine. In a near future, we will implement it as a secure proxy for publishing geo-data.

7 Related Work

Several access control models and approaches have been proposed for geo-spatial resources. Some of them such as the Geospatio-temporal Authorization Model (GSAM) focus on the visualization of raster geo-spatial data like multi-resolution satellite imagery (see [15], [16], [17] and [3] for details) while others like Geo-RBAC

[2, 18] may be described as Location Based Systems. On our side, we proposed an extension to the generic Or-BAC model to derive a geospatial context aware access control system ([4] and [5]). Regarding security standards, the Open Geospatial Consortium (OGC) [14] published the Digital Rights Management Reference Model (GeoDRM RM) [19] which is a reference model for digital rights management functionality for geospatial resources and geo-XACML [20] which extends the OASIS XACML [21] language for expressing authorization policies. The interested reader can refer to [22] for a summary of the current state of the art in the field of geo-spatial databases security.

As we already mentioned, to our knowledge it is the first time that a security model includes a framework for specifying protection mechanisms to be enforced. Most of the existing works on geo-data security focus on the expressive power of the security policy and on conflict resolution between permissions and prohibitions. This is the case in [23] where the authors, in the context of an XML-based Framework, propose to use Scalable Vector Graphics (SVG) [24] to represent geo-spatial objects and layers. They then define an access control model where an authorizations rule involves a subject, an object and an action as well as a Level of Details factor and an operative region. The SVG representation of the map and R-tree based indexes are used in the policy enforcement algorithm to determine which geo-spatial objects are addressed by the request and whether they can be accessed or not. In [25], authors assume that all spatial data are stored in a spatial database accessed by a Geographic Information System (GIS). A security object may be a spatial component, a set of spatial components or indirectly a query result. The algorithm which analyzes access requests includes a step of potential conflicts detection between security rules involving geo-spatial objects which can touch, intersect or be contained in each other. The authors distinguish between two potential cases of conflict depending on whether an object is totally or partially included in another. In [26], the author makes the distinction between object-based restrictions (on a particular object), class-based restrictions (on all objects of type “Building” or type “Road” for example) or spatial access restrictions (based on the geometry of objects). Objects are encoded using the Geographic MarkUp Language (GML) [27]. Security rules are expressed using XACML and geoXACML and may contain a spatial condition. Evaluation of the security policy may result in either “Permit”, “Deny”, “N/A” or “indeterminate”. The paper focuses on the “approximate” detection of contrary spatial permissions i.e. one spatial rule evaluates to permission while another one evaluates to prohibition. For this “approximate” detection, no actual request is required. The author states that a complex access control system has to ensure appropriate and error-free enforcement of declared permissions. He suggests using a permission repository and testing it for the a priori detection of inconsistent spatial authorization rules.

8 Conclusion

In this paper we focused on how to enforce the security policy in the framework of a Map Service supporting the creation and display of map-like views of information. We proposed a rule-based PEP which selects the protection mechanism to enforce

whenever a prohibition is derived from the security policy. We suggested seven protection mechanisms, namely: {*zoom-in*, *pixelizate*, *blur*, *mask*, *paste*, *hide*, *reject_query*}. We defined the logical framework to express some protection rules. These rules specify mechanisms to enforce whenever prohibitions are derived from the security policy. If, given a prohibition, several mechanisms could be used then only one of them should be selected according to priorities which are either assigned to the protection mechanisms themselves or to the protection rules. We defined the PEP algorithm which enforces the mechanisms and builds the map to display. We presented an example to illustrate how our proposal could be used and useful in a real application. We sketched the implementation of our proposal within the framework of the SLD profile of the WMS encoding standard and finally, we implemented a prototype showing the feasibility of our approach. Finally, let us also mention the following point: geographic data are a special case of multimedia data. Therefore, we also proposed a version of our model for the more generic case of multimedia data [28]. We used our model to protect images published in a social network.

Acknowledgment

This work was conducted as part of the ANR funded project under reference ANR-SESUR-2007-FLUOR.

References

1. 2011 WikiPedia. *Satellite map images with missing or unclear data*. http://en.wikipedia.org/wiki/Satellite_map_images_with_missing_or_unclear_data
2. Bertino, E., Catania, B., Damiani, M.L., Perlasca, P.: GEO-RBAC : A spatially Aware RBAC. ACM Symposium on Access Control Models and Technologies (SACMAT'05), Stockholm, Sweeden (2005) 29-37
3. Atluri, V., Chun, S.A.: A geotemporal role-based authorization system. International Journal of Information and Computer Security **1** (2007) 143-168
4. Gabillon, A., Capolsini, P.: Dynamic Security rules for Geo Data. In: Sciences, L.N.i.C. (ed.): International workshop on Autonomous and Spontaneous Security (SETOP'09), Vol. Lecture Notes in Computer Science 5939. LNCS 5939 - Springer-Verlag, St Malo, France (2009) 136-152
5. Capolsini, P., Gabillon, A.: Security policies for the Visualization of Geo Data. ACM SIGSPATIAL GIS 2009 International Workshop on Security and Privacy in GIS and LBS (SPRINGL'09). ACM, Seattle, WA, USA (2009) 02-11
6. Gabillon, A., Capolsini, P.: Rule-based Policy Enforcement Point for Map Services. ACM SIGSPATIAL GIS 2010 International Workshop on Security and Privacy in GIS and LBS (SPRINGL'10). ACM, San Jose, CA, USA (2010) 12-17
7. Lupp, M.: Styled Layer Descriptor profile of the Web Map Service Implementation Specification. Open Geospatial Consortium Inc. **OGC(R) 05-078r4** (2007)
8. Beaujardiere, J.d.l.: OpenGIS(R) Web Map Server Implementation Specification. Open Geospatial Consortium Inc. **OGC(R) 06-042** (2006)
9. El-Kalam, A., El-Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miège, A., Saurel, C., Trouessin, G.: Organization Based Access Control. 4th IEEE International Workshop on Policies for Distributed Systems and Networks (Policy'03). IEEE, Como, Italy (2003)

10. Yuan, E., Tong, J.: Attributed Based Access Control (ABAC) for Web Services. Proceedings of the IEEE International Conference on Web Services (ICWS'05), Orlando, Florida - USA (2005)
11. Janée, G., Frew, J., Hill, L.L.: Issues in Geo-referenced Digital Libraries. D-Lib Magazine, Vol. 10 (2004)
12. Rigaux, P., Scholl, M., Voisard, A.: Spatial Databases with application to GIS. Elsevier (2002)
13. Herring, J.R.: OpenGIS(R) Implementation Specification for Geographic information - Simple feature access - Part 1 : Common architecture. Open Geospatial Consortium Inc. **OGC(R) 06-103r3** (2006)
14. [OGC2008] OGC. *Open Geospatial Consortium Inc. - About Us*; http://www.opengeospatial.org/about.
15. Chun, S.A., Atluri, V.: Protecting privacy from continuous high-resolution satellite surveillance. In Proceedings of the 14th IFIP 11.3 Annual Working Conference on Database Security, Schoorl, The Netherlands (2000) 233-244
16. Atluri, V., Mazzoleni, P.: A uniform indexing scheme for geo-spatial data and authorizations. In Proceedings of the 16th IFIP WG 11.3 Conference on Data and Application Security (2002)
17. Atluri, V., Chun, S.A.: An authorization Model for Geospatial Data. IEEE Transactions on Dependable and Secure Computing **1** (2004) 238-254
18. Damiani, M.L., Bertino, E., Catania, B., Perlasca, P.: GEO-RBAC : A spatially Aware RBAC. ACM Transactions on Information Systems and Security **00** (2006) 1-34
19. Volwes, G.: Geospatial Digital Rights Management Reference Model (GeoDRM RM). Open Geospatial Consortium Inc. **OGC(R) 06-004r3** (2006)
20. Matheus, A., Herrmann, J.: Geospatial eXtensible Access Control Markup Language (GeoXACML). Open Geospatial Consortium Inc. **OGC(R) 07-026r2** (2008)
21. [XACML22005] OASIS. *eXtensible Access Control Markup Language (XACML) Version 2.0*; http://www.oasis-open.org.
22. Chun, S.A., Atluri, V.: Geospatial Database Security. In: Gertz, M., Jajodia, S. (eds.): Handbook of Database Security Applications and Trends. Springer US (2008) 247-266
23. Purevjii, B.-O., Amagasa, T., Imai, S., Kanamori, Y.: An access control model for geographic data in an XML-based framework. 2nd International workshop on Security in Information Systems (WOSIS'04), Porto, Portugal (2004) 251-260
24. [SVG2010] W3C. *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*; http://www.w3.org/Graphics/SVG/.
25. Sasaoka, L.K., Medeiros, C.B.: Access Control in Geographic Databases Advances in Conceptual Modeling - Theory and Practice (Lecture Notes in Computer Science) **4231/2006** (2006) 110-119
26. Matheus, A.: Declaration and enforcement of fine-grained access restrictions for a service-based geospatial data infrastructure. 10th ACM Symposium on Access Control Models and Technologies (SACMAT'05), Stockholm, Sweden (2005) 21-28
27. Portele, C.. OpenGIS(R) Geography Markup Language (GML) Encoding Standard. Open Geospatial Consortium Inc. **OGC(R) 07-036** (2007)

28. Bechara al Bouna, Richard Chbeir, Alban Gabillon: The Image Protector - A Flexible Security Rule Specification Toolkit . SECRYPT 2011: 345-350. Proceedings of the International Conference on Security and Cryptography, Seville, Spain, 18 - 21 July, 2011