Security Mechanisms for Geo-Data

Alban Gabillon, Patrick Capolsini

Security Mechanisms for Geographic data

Alban Gabillon
Université de la Polynésie française
BP 6570 Faa'a Aéroport
+689 80 38 83
alban.gabillon@upf.pf

Patrick Capolsini
Université de la Polynésie française
BP 6570 Faa'a Aéroport
+689 80 38 80
patrick.capolsini@upf.pf

ABSTRACT

In the framework of a map service which creates and displays maps of information coming from multiple heterogeneous sources, implementing a prohibition can be done in several ways. A sensitive object can be erased from the returned map, or masked, or blurred or even replaced by another object. 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.

Categories and Subject Descriptors

K.6.m [**Management of computing and information System**]: Miscellaneous – *Security*

General Terms

Security

Keywords

Access Control, Geo-spatial Data visualization, Maps service, Policy Enforcement Point

1. INTRODUCTION

Several models for geo-spatial data have already been proposed [1-4]. Most of these models consider 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. On our side in [3], 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 [4], 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. All these works focus on how to express geo-spatial security rules and on how to solve conflicts between them. In this paper we do not focus on the security policy anymore. We rather investigate the way the security policy can be enforced in the framework of a Map Service. Map Services support the creation and display of map-like views of information coming from multiple heterogeneous sources. In such a framework, denying access to a non authorized object can be done in several ways. Of course, the unauthorized object can simply be erased from the final map. However there are cases where this would not be relevant. For example, consider a river which crosses an unauthorized military area. Because, the river crosses this area, we derive from the security policy that access to the river should be denied. In such a case, it would not be realistic to completely erase the river from the final map. The best solution would be to erase the portion which crosses the military area. The Policy Enforcement Point (PEP) can do this by putting a mask on top of the military area. In the same way, consider a user requesting a map at a zoom-in factor of 5. The security policy says that this user is forbidden to see the map at such a zoom-in factor. However, the security policy says also that the same user is authorized to see the same map at a zoom-in factor of 4. Instead of rejecting the user query, the PEP can instead return the requested map at a zoom-in factor of 4. It is also sometimes more appropriate to blur an object instead of erasing it from the final map. This protection mechanism is used by Google Earth to protect sensitive areas like Dick Cheney's house [5] or the white house. These examples show us that in geographic applications, different protection mechanisms can be used to implement a prohibition. In this paper we suggest a rule-based PEP for Map services where Protection Mechanism (PM) rules define protection mechanisms that the PEP should enforce to implement prohibitions. To our knowledge, it is the first time that a security model includes a complete framework for specifying protection mechanisms.

In section 2 of this paper, we recall the basics of the security model we already defined in [3] and [4]. In section 3 we define PM 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 review related works. Finally section 7 concludes this paper.

2. SECURITY MODEL

In [3] and [4], we proposed a complete security model, based on the OrBAC model [6], for expressing security policies for geographic applications. Due to space limitations and since this paper focuses rather on policy implementation than on the

MEDES'10, October 26-20, 2010, Bangkok, Thailand.

security policy itself, we present here a simplified version of our model based on the ABAC model [7].We define geometric entities, spatial analysis functions, spatial predicates, spatial query and authorization rules.

2.1 Geometric Entities

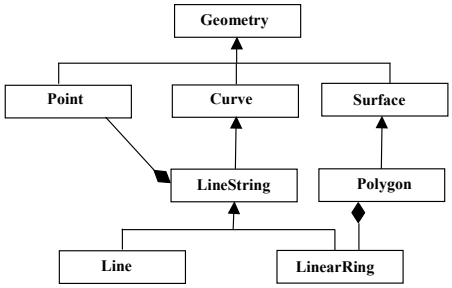

Fig. 1. OpenGIS Geometry Class Hierarchy

A *georeferenced (geometric) object* is a granule of information that is relevant to an identifiable subset of the Earth's surface [8]. Any geometric object has the following two components [9] : 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 [10]. In this paper, we do not consider the whole class hierarchy defined in [10]. For the sake of simplicity, we consider only the branch depicted in figure 1.

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*, [10] defines eight spatial predicates namely, *Equals*, *Disjoint*, *Intersects*, *Touches*, *Crosses*, *Within*, *Contains* and *Overlaps*

- $\forall g_1, \forall g_2, Equals(g_1, g_2) \leftrightarrow (g_1 \cap g_2) = (g_1 \cup g_2)$
  $\forall g_1, \forall g_2, Disjoints(g_1, g_2) \leftrightarrow g_1 \cap g_2 = \varnothing$
  $\forall g_1, \forall g_2, Touches(g_1, g_2) \leftrightarrow$
  $(I(g_1) \cap I(g_2) = \varnothing) \wedge (g_1 \cap g_2 \neq \varnothing)$
- $\forall g_1, \forall g_2, Crosses(g_1, g_2) \leftrightarrow$
  $(dim(I(g_1) \cap I(g_2)) < max(dim(g_1), dim(g_2)))$
  $\wedge (g_1 \cap g_2 \neq g_1) \wedge (g_1 \cap g_2 \neq g_2)$
- $\forall g_1, \forall g_2, Within(g_1, g_2) \leftrightarrow$
  $(g_1 \cap g_2 = g_1) \wedge (I(g_1) \cap E(g_2) \neq \varnothing)$
- $\forall g_1, \forall g_2, Contains(g_1, g_2) \leftrightarrow Within(g_2, g_1)$
- $\forall g_1, \forall g_2, Overlaps(g_1, g_2) \leftrightarrow$
  $(dim(I(g_1)) = dim(I(g_2)) = dim(I(g_1) \cap I(g_2)))$
  $\wedge (g_1 \cap g_2 \neq g_1) \wedge (g_1 \cap g_2 \neq g_2)$
- $\forall g_1, \forall g_2, Intersects(g_1, g_2) \leftrightarrow \neg Disjoint(g_2, g_1)$

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 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:

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

Permit(s,o) reads "s is permitted to display object o."

We define a negative authorization rule as a logical rule having the following form:

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

Deny(s,a,o) reads "s is forbidden to display object o."

Let us consider the following example of security policy which consists of the four following rules:

The first security rule says that civilians cannot display tanks.

$$\forall s \forall o (Civilian(s) \wedge Tank(o) \rightarrow Deny(s, o))$$

The second security rule says that civilians cannot display barracks at a zoom-in factor greater than 1.

$$\forall s \forall o \begin{pmatrix} Civilian(s) \wedge Barrack(o) \wedge zf(o) > 1 \\ \rightarrow Deny(s,o) \end{pmatrix}$$

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

$$\forall s \forall o \big(Soldier(s) \wedge Tank(o) \rightarrow Permit(s,o) \big)$$

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

$$\forall s \forall o \begin{pmatrix} Soldier(s) \wedge Tank(o) \wedge \\ \neg Within(o, MilitaryZone) \rightarrow Deny(s,o) \end{pmatrix}$$

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 [3], 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 aim, in this paper, is to provide us with 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. PROTECTION MECHANISM RULES

In this section, we define a complete framework for specifying the protection mechanism which should be enforced in case a user is denied to display a given object.

A *Protection Mechanism (PM) rule* is a rule of the form:

$$\forall s \forall o \big(Condition \wedge Deny(s,o) \rightarrow Protect(o,M) \big)$$

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"
- *blur* : reads "object *o* should be blurred in the map"
- *mask(g)*: reads "insert mask *g* in the map"
- *erase*: reads "erase *o* from the map"
- *replace(g)*: reads "replace *o* with *g* in the map"
- *zoom_in(i)*: reads "forces the zoom-in factor of object *o* to a value which is less than or equal to *i*"

Let us consider the following three examples of PM rules:

$$\forall s \forall o \begin{pmatrix} Civilian\ (s) \wedge Tank\ (o) \wedge Deny\ (s,o) \\ \rightarrow Protect\ (o, reject\ \_ query\ ) \end{pmatrix}$$

$$\forall s \forall o \begin{pmatrix} Soldier\ (s) \wedge Tank\ (o) \wedge Deny\ (s,o) \\ \rightarrow Protect\ (o, erase\ ) \end{pmatrix}$$

$$\forall s \forall o \begin{pmatrix} Civilian\,(s) \wedge Barrack\,(o) \wedge Deny\,(s,o) \\ \rightarrow Protect\,(o, zoom\ \_ in(1)) \end{pmatrix}$$

The first rule says that if civilians who are forbidden to see tanks request to see them then their query should simply be rejected i.e. empty map should be returned. The second rule says that if soldiers who are forbidden to see tanks request to see them then tanks should be erased from the returned map. The third rule says that civilians who are 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 PM 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*.

$$\forall s \forall o \big(Deny\ (s,o) \rightarrow Protect\ (o, blur\ ) \big)$$

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, blur, mask, erase, replace, reject_query*},
- or we assign priorities to PM rules (with the default rule having the lowest priority).

In the next section we design a PEP algorithm which works with both solutions.

4. PEP ALGORITHM

In this section we define an algorithm for (i) enforcing protection mechanisms and (ii) construct the map to display. This algorithm works in the following two cases:

- Mechanisms have different priorities and the following mechanisms hierarchy is used (from the lowest priority to the highest priority): {zoom-in, blur, mask, erase, replace, 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 PM rules (with the default rule having the lowest priority). Thanks to these priorities, only one mechanism can be derived from a single prohibition and the algorithm enforces it. However, it might happen that several mechanisms can be derived from a single prohibition. This can be the case if several PM 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 and an empty map is returned.
- For the sake of simplicity, we assume that mechanisms are mutually exclusive. We cannot have two different mechanisms applying to the same object. It would be however, perfectly possible to design an algorithm allowing us to protect one object with several different mechanisms (e.g. *replace(g)* and then *blur*).
- The returned map is displayed at the lowest zoom-in factor imposed by PM rules. 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 at respectively zoom-in factor equal to 4 and zoom-in factor equal to 3. The lowest zoom-in factor imposed by PM rules is selected and the map is displayed at zoom-in factor equal to 3.

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(g,map)* denotes a procedure which inserts geo-referenced object *g* into map *map*. *blur(o)* denotes a function which blurs *o*. *applyzf(i,map)* is a function which applies zoom-in factor *i* on map *map*

```
map ← empty_map
minzf ← zf(q)
For o in O(q)
 If Protect(o,reject_query) then
  return(empty_map)
 Else
  If  Protect(o,replace(g)) then
   insert(g,map)
  Else
   If NOT Protect(o,erase) then
    If Protect(o,mask(g)) then
     insert(g,map)
    Else
     If Protect(o,blur) then
      insert(blur(o),map)
     Else
      If Protect(o,zoom_in(i)) then
         minzf ←min(i,minzf)
      insert(o,map)
return(applyzf(minzf,map))
```

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

This section is based on the example of security policy we already defined in [3]. We first resume the main lines of this security policy and then we define some Protection Mechanisms rules specifying which mechanisms should be used to enforce prohibitions.

5.1 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 2 shows that subjects are drivers who can be either taxi drivers or ambulance drivers. Objects are roads which can be either main roads or secondary roads, hospitals (including military hospitals) and gas stations. Basically, the security policy expresses the fact that drivers can display spatial data which are within a radius of 40 km around their position. However, there are some restrictions to this general rule.

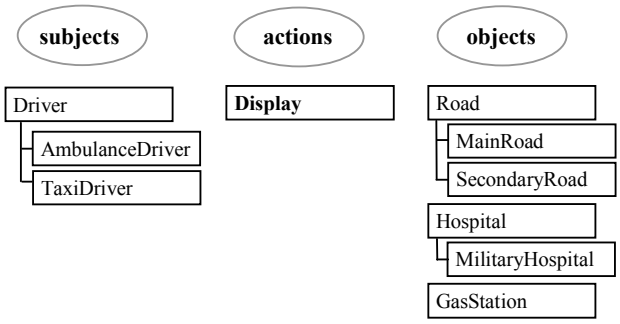

Fig. 2. Synopsis of our example

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.

Rule 1: Drivers have the permission to display at a maximum zoom-in factor of 5 any object that is within a radius of 40km around their position,

$$\forall s \forall o \left(\begin{array}{l} Driver(s) \wedge distance(s,o) \leq 40 \wedge zf(o) \leq 5 \\ \quad \rightarrow Permit(s,o) \end{array} \right)$$

Rule 2: Drivers have the permission to display main roads with a maximum zoom-in factor of 2:

$$\forall s \forall o \left(\begin{array}{l} Driver(s) \wedge MainRoad(o) \wedge zf(o) \leq 2 \\ \quad \rightarrow Permit(s,o) \end{array} \right)$$

Note that if the subject is a driver and the object is a main road located within a radius of 40km then rule 1 says that drivers can display the main road at a zoom-in factor of 5.

Rule 3: Drivers driving at a speed greater than 100km per hour are forbidden to display any object.

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

Rule 4: Gas stations which are not on the way cannot be displayed.

$$\forall s \forall o \left(\begin{array}{l} Driver(s) \wedge GasStation(o) \wedge \\ \neg \left(\exists r, Road(r) \wedge Within(s,r) \wedge Touches(o,r) \right) \\ \quad \rightarrow Deny(s,o) \end{array} \right)$$

Rule 5: Finally taxi drivers are prohibited to display military hospitals at a zoom-in factor greater than 4

$$\forall s \forall o \left(\begin{array}{l} TaxiDriver(s) \wedge MilitaryHospital(o) \wedge zf(o) > 4 \\ \quad \rightarrow Deny(s,o) \end{array} \right)$$

The above security policy may lead to conflicts. Rule 1 and rule 2 conflict with the default policy. If the subject drives at more than 100 km per hour then rule 3 conflicts with rule 1 and rule 2. For gas stations which are not on the way, rule 4 conflicts with rule 1 and rule 2. For military hospitals and taxi drivers, rule 5 conflicts with rule 1 if the zoom-in factor of the query is 5. As we said before, it is not the purpose of this paper to discuss conflict resolution. However let us mention that rules 3, 4 and 5 are seen as exceptions to rule 1 and rule 2.

5.2 Protection Mechanisms Rules

Default mechanism: We define the default mechanism as blur:

$$\forall s \forall o \left(Deny\ (s,o) \rightarrow Protect\ (o, blur\ ) \right)$$

In our example, this rule will apply in particular to instances of the Deny predicate which are derived from the default (closed) policy. These instances address objects which are not main roads and which are outside a radius of 40km.

Deny derived from rule 3: Drivers whose speed is greater than 100km per hour and who are forbidden to display objects should see their query rejected i.e. drivers are in fact forbidden to use the application as long as they drive fast.

$$\forall s \forall o \begin{pmatrix} Driver(s) \wedge speed(s) \geq 100 \wedge Deny(s,o) \\ \rightarrow \text{Protect}(o, reject\_query) \end{pmatrix}$$

Deny derived from rule 4: Subjects who are forbidden to display gas stations should not see them at all on the returned map.

$\forall s \forall o \left(GasStation(o) \wedge Deny(s,o) \rightarrow \text{Protect}(o, erase) \right)$ This rule applies not only to instances of the Deny predicate derived from rule 4 but also to instances of the Deny predicate which are derived from the default (closed) policy. These instances address gas stations which are outside the radius of 40km.

Deny derived from rule 5: Subjects who are forbidden to display military hospitals should be able to see them at a zoom-in factor which is less than or equal to 4.

$$\forall s \forall o \begin{pmatrix} MilitaryHospital(o) \wedge Deny(s,o) \rightarrow \\ \text{Protect}(o, zoom-in(4)) \end{pmatrix}$$

This rule applies to instances of the Deny predicate addressing taxi drivers. Indeed, unlike ambulance drivers, they are forbidden to display military hospitals at a zoom-in context greater than 4.

Note that, if priorities are assigned to mechanisms and `blur` (default mechanism) has a higher priority than `zoom-in` then this rule will however never be active.

6. RELATED WORKS

Several access control models and approaches have been proposed for geo-spatial resources. Some of them like the Geospatio-temporal Authorization Model (GSAM) focus on the visualization of raster geo-spatial data like multi-resolution satellite imagery (see [11], [12], [13] and [2] for details) while others like Geo-RBAC [1, 14] 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 ([3] and [4]). Regarding security standards, the Open Geospatial Consortium (OGC) [15] published the Digital Rights Management Reference Model (GeoDRM RM) [16] which is a reference model for digital rights management functionality for geospatial resources and geo-XACML [17] which extends the OASIS XACML [18] language for expressing authorization policies. The interested reader can refer to [19] for a summary of the current state of the art in the field of geo-spatial databases security.

As we already mentioned, it is the first time, to our knowledge, that a security model includes a framework for specifying protection mechanisms to be enforced. Most 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 [20] where the authors, in the context of an XML-based Framework, propose to use Scalable Vector Graphics (SVG) [21] 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 [22], 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 [23], 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) [24]. 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.

7. 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 six protection mechanisms, namely: *reject_query*, *erase, replace, mask, blur* and *zoom-in*. We defined the logical framework to express some Protection Mechanisms 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 PM rules. We defined the PEP algorithm which enforces the mechanisms and builds the map to display. Finally, we presented a small example to illustrate how our proposal could be used and useful in a real application. We are currently studying the cost of deriving instances of the *Protect* predicate and working on a complete implementation of our proposal in the framework of the OGC® Web Map service.

8. Acknowledgments

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

9. References

[1] Bertino, E., et al. *GEO-RBAC : A spatially Aware RBAC*. in *ACM Symposium on Access Control Models and Technologies (SACMAT'05)*. 2005. Stockholm, Sweeden.p. 29-37

[2] Atluri, V. and S.A. Chun, *A geotemporal role-based authorization system.* International Journal of Information and Computer Security, 2007. **1**(1/2): p. 143-168.

[3] Gabillon, A. and P. Capolsini. *Dynamic Security rules for Geo Data*. in *International workshop on Autonomous and Spontaneous Security (SETOP'09)*. 2009. St Malo, France: LNCS 5939 - Springer-Verlag.p. 136-152

[4] Capolsini, P. and A. Gabillon. *Security policies for the Visualization of Geo Data*. in *ACM SIGSPATIAL GIS 2009 International Workshop on Security and Privacy in GIS and LBS (SPRINGL'09)*. 2009. Seattle, WA, USA: Association for Computing Machinery.p. 02-11

[5] Weinberger, S., *Why is Google Earth Hiding Dick Cheney's House?*, in *http://www.wired.com*. 2008.

[6] El-Kalam, A., et al. *Organization Based Access Control*. in *4th IEEE International Workshop on Policies for Distributed Systems and Networks (Policy'03)*. 2003. Como, Italy: IEEE.p.

[7] Yuan, E. and J. Tong. *Attributed Based Access Control (ABAC) for Web Services*. in *Proceedings of the IEEE International Conference on Web Services (ICWS'05)*. 2005. Orlando, Florida - USA.p.

[8] Janée, G., J. Frew, and L.L. Hill, *Issues in Geo-referenced Digital Libraries*, in *D-Lib Magazine*. 2004.

[9] Rigaux, P., M. Scholl, and A. Voisard, *Spatial Databases with application to GIS*. 2002: Elsevier. 410.

[10] Herring, J.R., *OpenGIS(R) Implementation Specification for Geographic information - Simple feature access - Part 1 : Common architecture.* Open Geospatial Consortium Inc., 2006. **OGC(R) 06-103r3**.

[11] Chun, S.A. and V. Atluri. *Protecting privacy from continuous high-resolution satellite surveillance*. in *In Proceedings of the 14th IFIP 11.3 Annual Working Conference on Database Security*. 2000. Schoorl, The Netherlands.p. 233-244

[12] Atluri, V. and P. Mazzoleni. *A uniform indexing scheme for geo-spatial data and authorizations*. in *In Proceedings of the 16th IFIP WG 11.3 Conference on Data and Application Security*. 2002.p.

[13] Atluri, V. and S.A. Chun, *An authorization Model for Geospatial Data.* IEEE Transactions on Dependable and Secure Computing, 2004. **1**(4): p. 238-254.

[14] Damiani, M.L., et al., *GEO-RBAC : A spatially Aware RBAC.* ACM Transactions on Information Systems and Security, 2006. **00**(00): p. 1-34.

[15] OGC. *Open Geospatial Consortium Inc. - About Us*. 2008 [cited; Available from: http://www.opengeospatial.org/about.

[16] Volwes, G., *Geospatial Digital Rights Management Reference Model (GeoDRM RM).* Open Geospatial Consortium Inc., 2006. **OGC(R) 06-004r3**.

[17] Matheus, A. and J. Herrmann, *Geospatial eXtensible Access Control Markup Language (GeoXACML).* Open Geospatial Consortium Inc., 2008. **OGC(R) 07-026r2**.

[18] OASIS. *eXtensible Access Control Markup Language (XACML) Version 2.0.* 2005 [cited; Available from: http://www.oasis-open.org.

[19] Chun, S.A. and V. Atluri, *Geospatial Database Security*, in *Handbook of Database Security Applications and Trends*, M. Gertz and S. Jajodia, Editors. 2008, Springer US. p. 247-266.

[20] Purevjii, B.-O., et al. *An access control model for geographic data in an XML-based framework*. in *2nd International workshop on Security in Information Systems (WOSIS'04)*. 2004. Porto, Portugal.p. 251-260

[21] W3C. *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*. 2010 [cited; W3C:[Available from: http://www.w3.org/Graphics/SVG/.

[22] Sasaoka, L.K. and C.B. Medeiros, *Access Control in Geographic Databases* Advances in Conceptual Modeling - Theory and Practice (Lecture Notes in Computer Science), 2006. **4231/2006**: p. 110-119.

[23] Matheus, A. *Declaration and enforcement of fine-grained access restrictions for a service-based geospatial data infrastructure*. in *10th ACM Symposium on Access Control Models and Technologies (SACMAT'05)*. 2005. Stockholm, Sweden.p. 21-28

[24] Portele, C., *OpenGIS(R) Geography Markup Language (GML) Encoding Standard.* Open Geospatial Consortium Inc., 2007. **OGC(R) 07-036**.