Geospatial eXtensible Access Control Markup Language

Andreas Matheus, Universität der Bundeswehr München
andreas.matheus(at)unibw.de

MOTIVATION

Controlling the access and the use of high quality and up-to-date geographic information is an important concern to many data providers. Possible application scenarios, requiring protection of different kinds can vary from data mining to data production and data use in a Spatial Data Infrastructure.

For the establishment of access protection that shall support various application scenarios, the flexible declaration of access rights and their enforcement is very important. Here, use of standards can ensure interoperability and guarantee trusted access protection by mature security components. The interoperability aspect is in particular important for the harmonization or creation of access rights across jurisdictions.

Just think of user Joe in a Spatial Data Infrastructure who relies on combining protected maps and demographic data avail-able from different providers, where incompatible security systems are used to enforce access rights. The advantage of using OGC's approved Web Map Service or Web Feature Services is no longer relevant, as different in-house security solutions potentially break that interoperability.

The Geospatial eXtensible Access Control Markup Language (GeoXACML) 1.0, a standard by the Open Geospatial Consortium, defines a Policy Language for the declaration and enforcement of access rights for distributed geographic information. GeoXACML is an extension of the eXtensible Access Control Markup Language (XACML), a standard by OASIS. The major strength of GeoXACML is that it allows the declaration and maintenance of access rights in an extremely flexible way, adapted to the data model of the geographic information that is to be protected.

This paper introduces the geo-specific Policy Language GeoXACML and illustrates its use by showing an example based on a simple architecture for protecting access to OGC's famous Web Map Service. The paper concludes with the discussion of requirements for structuring and transporting security information from the WMS client to the service using OASIS's Security Markup Language (SAML). In the outlook the author indicates implications to OGC's standardization process to enable protection for OpenGIS Web Services.

INTRODUCTION TO XACML

In the general domain of Access Control for a Service Oriented Architecture it is very important, to declare and enforce almost any right that you can imagine. In order to do so, the administrator of rights has to rely on software that is absolutely error free and does exactly regulate the access, as expressed in the rights. This can be achieved quite easily for right structures that are fairly simple or have been around for many years. One good example is the mature access control scheme for the UNIX file system that we are all familiar with.

In a distributed environment however, where access shall be controlled to a web service and the information it is serving, that scheme is not flexible enough. Therefore, different companies joined the Organization for the Advancement of Structured Information Standards (OASIS) (see [4]) to explore a language for writing flexible access rights that can be interpreted by software. The result is the eXtensible Access Control Markup Language (XACML) (see [5]), which is currently available at version 2.0. The standard provides multiple pieces of information from which the following is important in this context:

- the structure and the XML elements of the XACML Policy Language itself and the required schemata for validating policy instances,
- the semantics of the elements of the Policy Language and
- the rules, how to derive an authorization decision based on a XACML Policy instance and a given authorization decision request

The UML model in Figure 1 below illustrates the XACML Policy Language structure. Most important to notice is that each <PolicySet>, <Policy> and <Rule> element has associated a <Target> element. This <Target> element contains simple matching functions for the Subject, Action and Resource as it is common to any Access Control System. In addition it provides matching capabilities for environment information thru the <Environment> element. This allows a powerful indexing of different <Rule> elements which are the actual drivers of an authorization decision. A <Rule> element may contain the optional <Condition> element which can host any complex constraint as a Boolean expression. Here, you can find the real power of XACML!

For deriving an authorization decision, the value of the <Effect> element from all matching <Rule> elements will be incorporated and “compressed” by all Rule- and Policy Combining Algorithms which are attributes of the <Rule> and <Policy> elements. The “compression” of effect values stops at the top most <PolicySet> element and the resulting value represents the authorization decision. Even though the Effect value of each <Rule> element can only be “Permit” or “Deny”, the final value can also be “Indeterminate” and “Not Applicable”. In order to determine if the authorization decision was processed with no errors, the authorization decision result contains the processing status information.

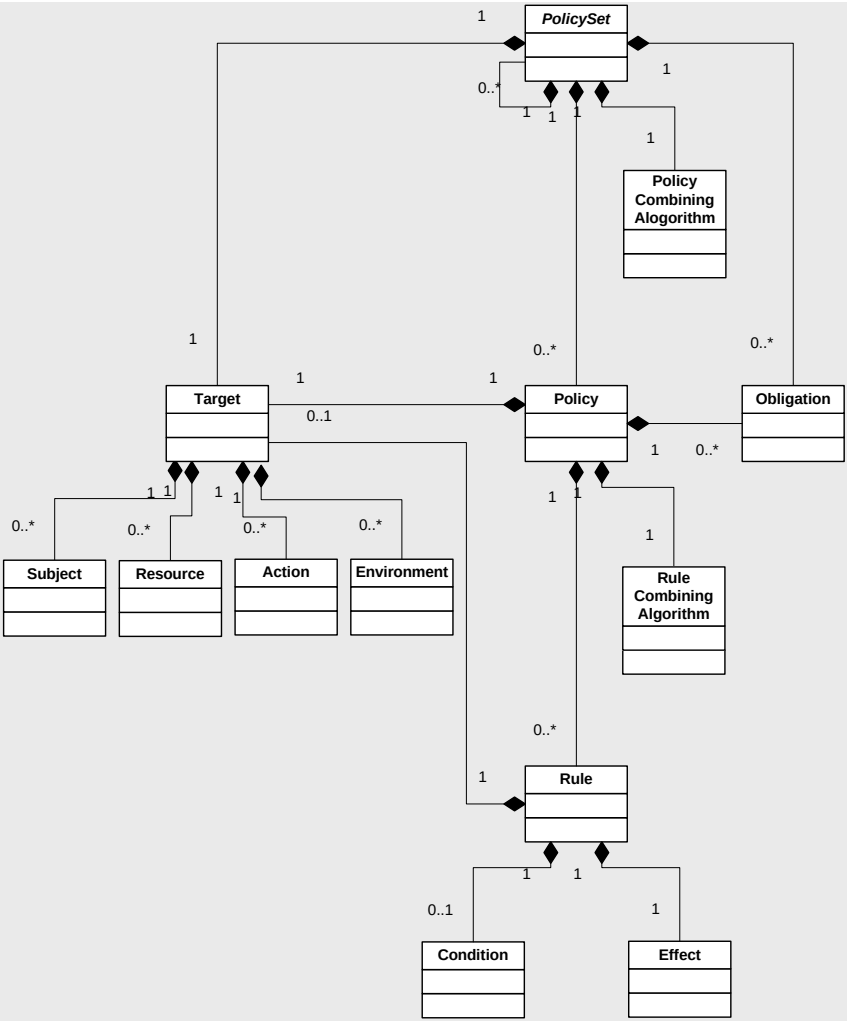

Figure 1: The XACML Policy Structure

The input for the authorization process is (i) the XACML Policy itself and (ii) the XACML Authorization Decision Request.

The XACML Authorization Decision Request contains the information about Subject, Action and Resource and optionally about the Environment upon which the authorization decision is to be derived. For a Service Oriented Architecture that information can be typically collected from the service request plus additional information such as IP-address of the client, actual date and time, used protocol such as HTTP or HTTPS, etc.
XACML foresees two possible ways of packaging the collected information into the actual Authorization Decision Request (see schema): (i) <AttributeValue> elements or (ii) XML document inserted into the <ResourceContent> element.

- Using the <AttributeValue> mechanism results in key-value-pairs, where the keys and their data type are pre-defined by the XACML standard. The information about the Subject and the Action should be encoded this way to allow indexing of <Rule> elements. This is recommended because it increases performance

of the authorization process. The values are obtained by using the <AttributeDesignator> element in the Policy.

- Using the <ResourceContent> mechanism requires the Policy Administrator to write conditions, where the value is fetched by using XPath expressions. The <Attribute-Selector> element provides the functionality to do so. In regard to performance of the authorization decision process, this mechanism is slower because XPath matching involves DOM processing. Building and evaluating the DOM for large XML documents, requires memory and processing power. However, this approach is much more flexible than the “simpler” <AttributeValue> approach.

Now that we have the information from the authorization decision request and the information from the Policy, the actual process of deriving the authorization decision can begin. In order to understand the mechanism, let’s see a Policy as a graph. The <PolicySet> and <Policy> elements represent the nodes and the <Rule> elements the leaves of the tree. Each edge contains matching information from the <Target> element. Based on that simple tree model, the authorization decision is processed in two phases: (i) Matching and (ii) Reasoning.

- The Matching phase starts at the root element. Here, the value of the combing algorithm controls which node(s) to process next and in which order. The set of all reached nodes and leaves represent one particular tree that is cut out of the Policy graph based on the information from an authorization decision request plus the matching and processing instructions from the policy.
- The Reasoning phase uses the previously created tree and starts the processing at the. The effect values from the leaves are used to resolve a final value at the root node by taking the combining algorithm processing instructions from each node under consideration.

Even though XACML provides a very powerful and very flexible way of declaring and deriving access control decisions, it does not provide any matching functions or data types that allow the reasoning of authorization decisions for geographic resources. However, XACML provides extensibility points that can be used for defining domain specific data types and functions. This is exactly what GeoXACML does!

INTRODUCTION to GeoXACML

GeoXACML 1.0 is a standard by the Open Geospatial Consortium (OGC) (see [1]) that is based on XACML 2.0. The requirement was to support the flexible declaration and enforcement of geo-specific access rights for geographic information, served by geo web services. In order to accomplish this, GeoXACML defines the appropriate geometric data-types and functions according to the core & extension meta-model. It uses the extensibility options from XACML to define a geometry data type and the relevant geo-specific functions as defined in the "Simple Features" specification by the OGC.

The GeoXACML 1.0 standard consists of three documents: The "core" document (OGC #07-026r2) (see [2]) defines the geometry data type and the functions as well as conformance classes for the functions. The two extensions define the allowed XML encoding of the supported geometry data types: Extension A (OGC #07-098r1) (see [3]) provides support for the GML2 geometry encoding and extension B (OGC #07-099r1) (see [4]) provides support for the GML3 geometry encoding.

The functions that are defined by the GeoXACML 1.0 standard are separated into different categories:

- **Topological Functions**: Equals, Disjoint, Touches, Crosses, Within, Contains, Overlaps, Intersects
- **Geometric Functions**: Buffer, Boundary, Union, Intersection, Difference, SymDifference, Centroid, ConvexHull, IsSimple, IsClosed, IsValid
- **Scalar Geometric Functions**: Distance, IsWithinDistance, Length, Area
- **Bag Functions**: GeometryOneAndOnly, GeometryBagSize, GeometryIsIn, GeometryBag
- **Set Functions**: GeometryBagIntersection, GeometryAtLeastOneMemberOf, GeometryBagUnion, GeometryBagSubset, GeometrySetEquals
- **Conversion Functions**: ConvertToMetre, ConvertToSquareMetre

Topological Functions allow the declaration and enforcement of geo-specific access rights where the geometries of two or more resources are compared. For example, "access is permitted if the requested map is within the bounding box (0,0 10,10)". Here, the box geometry could be static in the policy. Another example would be that "access is permitted if all requested land parcels are touching each other". Here, all geometries are retrieved from the <ResourceContent> XML document during the authorization decision process. So no geometries are declared in the policy itself!

Geometric Functions allow the declaration and enforcement of access rights in such a sense that the resulting geometries (that are constructed during the authorization process) fulfil certain characteristically constraints. For example, a right could use the convex hull function in the following manner: "access is permitted if the convex hull of all requested features of type A does not intersect with the convex hull of all requested features of type B". Or, "a user is permitted to insert features into a database if all geome-

tries are valid”. You could also declare this the negative way as “a user is denied … if any geometry is invalid”.

Scalar Geometric Functions can be used to express geo-specific access rights that rely on specific characteristics of resource geometries. For example: “access is permitted if the distance of all requested features is less than 1km from your current position”.

Bag and Set Functions allow comparing and constructing collections of geometries. So these functions support to express rights such as “access is denied if one of the requested features is in the set of restricted features”. This can be used to enforce rights that are dependent on classified resources.

Conversion Functions are relevant if the Policy Administrator uses Scalar Geometric Functions (which return metre or square metre) and does express constraints using non metric values. So for example, if the administrator likes to express the above restriction “… less than 1km” using miles “… less than 1mi” he can do so, but has to do the proper conversion! This is, because the distance function will return the value in metre.

In addition, the GeoXACML standard defines two conformance classes: Basic and Standard. This allows an implementation of a Policy Decision Point (PDP) to claim Basic or the Standard conformance by implementing the relevant functions plus support for either extension A and/or extension B geometry encoding. So an implementation of a PDP the claims conformance towards Basic / Extension A must implement all functions from the categories Topological, Bag, Set, Miscellaneous and Conversion Functions and support the GML2 geometry encoding.

EXAMPLE ARCHITECTURE FOR PROTECTING AN OGC WEB MAP SERVICE

Establishing an Access Control System for a Service Oriented Architecture can basically be achieved in two different ways: (i) modification of the web services software by integrating the relevant access control functionality or (ii) putting facades in front of the web services and let the façade deal with the access control issues.

In the geospatial domain, where Open GIS Web Services are very popular, different Open Web Services initiatives of the OGC have successfully proven that the “façade technique” is practical up to a certain point. For this paper, it is also good practice to use façades because it allows to clearly separate between the regular OGC service functionality and the “extra” functionality required for Access Control.

So for the purpose of this paper, the author likes to introduce an “OGC architecture extension” as it is proposed by the XACML standard. The XACML information flow, as shown in figure 2 below, introduces a concept to be implemented as a distributed Access Control System for Service Oriented Architectures.

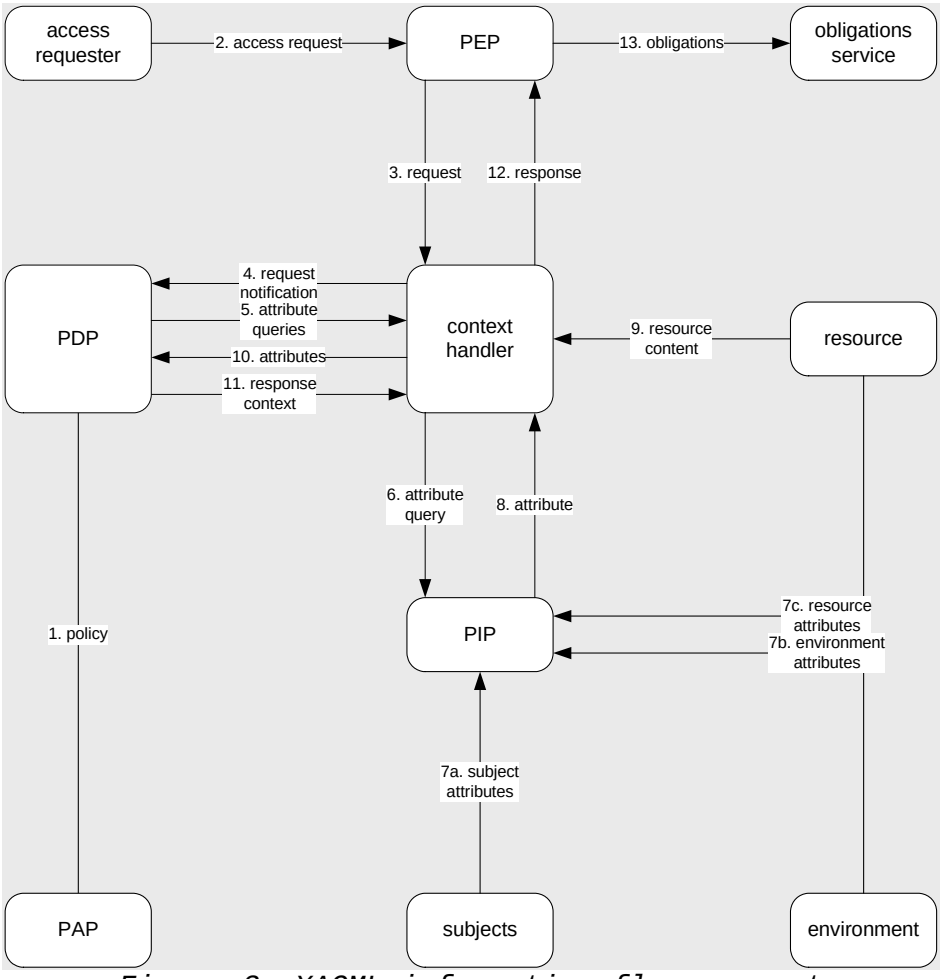

Figure 2: XACML information flow concept

Without detailed explanation, it is important to notice that there is a PEP, a context handler and a PDP.

The PEP (Policy Enforcement Point) is the software component that intercepts the requests to the service that is to be protected. It forwards the fetched information to the context handler, which creates the XACML Authorization Decision Request based on that information plus additional information and forwards it to the Policy Decision Point (PDP).

The PDP uses the request to derive the corresponding authorization decision by consulting the XACML Policy.

Adopting the XACML information flow model for protecting access to an OGC Web Map Service can be done by using the PEP as the façade to the WMS. It intercepts the WMS request from the client and creates the XACML Authorization Decision Request, send to the PDP. So the context handler is incorporated into the PEP.

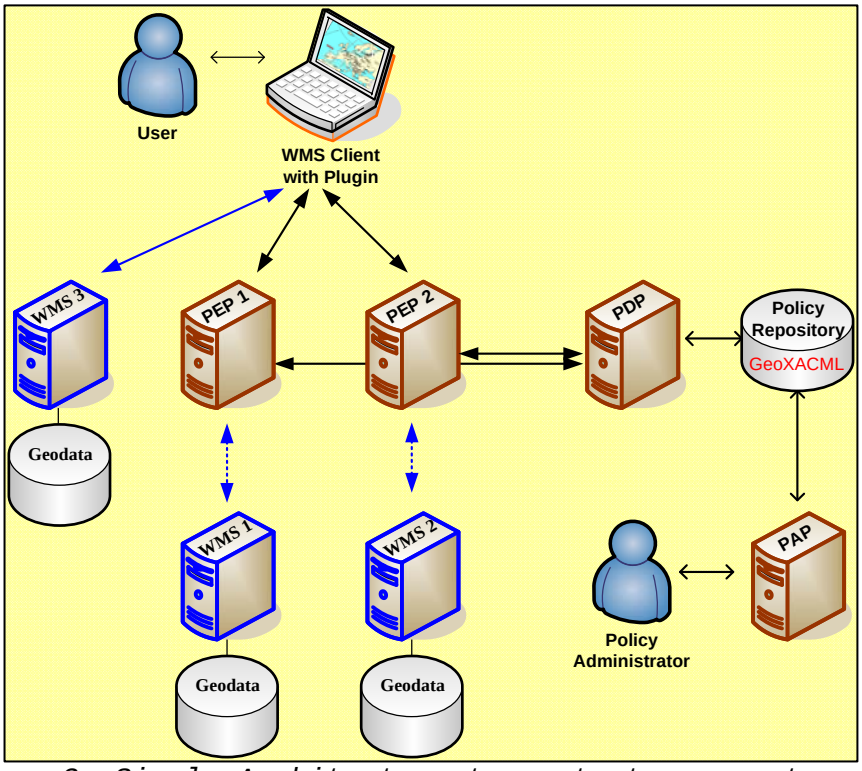

Figure 3: Simple Architecture to protect access to a WMS

In figure 3, the administration of the XACML policies is fulfilled by the policy administrator using the PAP. This service supports the declaration of access rights in a XACML policy but has nothing to do at runtime.

The above architecture is based on a single PDP that serves authorization decision responses to multiple PEPs. This is quite feasible but requires a trusted relationship between the PEPs and the PDP. In cases, where the PDP is provided by an external service provider, adequate security measures must be in place. For further reading, the interested reader is advised to the “Trusted Geo Services IPR” that is available as an OGC Discussion Paper (see [6]).

EXAMPLE GeoXACML POLICY FOR DECLARING ACCESS RIGHTS

Now that we have explored a possible architecture to protect access to a WMS, we can actually look at an example how to use GeoXACML to declare relevant access rights. The author likes to use the example from the www.geoxacml.org web site.

In this example, a user can combine maps that are received from two different Web Maps Services. The "demis WMS" is not protected and provides the background map. For the "Intergraph WMS", the layer CAPITALS is protected in the following sense:

> *Alice has the right to request maps (using the GetMap operation) from the "Intergraph WMS" of the layer CAPITALS for any area except America. She can also request feature information (using the GetFeatureInfo operation) for an area of Europe.*

The "kernel" part of the GeoXACML Policy that expresses Alice' geo-specific rights are based on topological functions, as illustrated in the following policy snippet (the full GeoXACML policy is available at the GeoXACML homepage):

```
<Function FunctionId="urn:ogc:def:function:geoxacml:1.0:geometry-contains"/>
<AttributeValue DataType="urn:ogc:def:dataType:geoxacml:1.0:geometry">
      <gml:Polygon gid="North and South America"
            srsName=" http://www.opengis.net/gml/srs/epsg.xml#4326"
            xmlns:gml="http://www.opengis.net/gml">
            <gml:outerBoundaryIs><gml:LinearRing>
                  <gml:coordinates cs="," ts=" ">
                  -180,60 -180,47 -137,55 -125,35 -110,17 -80,5 -87,-5
                  -74,2 -78,-53 -67,-58 -60,-48 -35,-23 -27,-3 -55,12 -60,22
                  -75,30 -67,42 -50,45 -60,85 -60,85 -85,85 -120,80 -130,75
                  -160,75 -168,70 -180,60
                  </gml:coordinates>
            </gml:LinearRing></gml:outerBoundaryIs >
      </gml:Polygon>
</AttributeValue>
<AttributeSelector RequestContextPath="//gml:boundedBy/gml:Box"
      DataType="urn:ogc:def:dataType:geoxacml:1.0:geometry"/>
```

Listing 1: Use of a Topological Function to express a geo-specific constraint as part of Alice' access rights

The snippet above is based on GML2 geometry encoding that defines a rough estimate approximation of the outer boundary of the American continent. This geometry is part of the GeoXACML Policy. The area, for which Alice is requesting a map is obtained from the XPath expression "//gml:boundedBy/gml:Box" applied to the XML encoded WMS request, available within the <ResourceContent> element of the XACML Authorization Decision Request (see previous section for details). So upon deriving the authorization decision, the geometry from the policy (<AttributeValue>) and the geometry of the bounding box of the GetMap request are compared using the topological function "contains". As you can not see from the snippet, this condition is one integral part of the <Rule> that defines Alice' rights and has the effect "Deny".

CONCLUSION AND OUTLOOK

This paper introduced the GeoXACML Policy Language – a standard by the Open Geospatial Consortium - and illustrated the use of GeoXACML in a simple example that protects access to a Web Map Service. The author likes to point out explicitly that GeoXACML is not limited to protect access to a WMS in particular or OGC Web Services only! It can be applied in many other circumstances where access to information with geographic characteristics is to be protected.

This paper further introduces a Service Oriented Architecture example that allows protecting a WMS through the "façade technique" maintaining OGC Web Services interoperability. Please note that in order to actually use this architecture in a productive environment; the user's identity has to be transported from the client to the protected service. This can be obtained in many ways. One possibility is to use the SSO Browser Profile (see [8]) from the Security Assertion Markup Language (SAML) (see [7]), a standard by OASIS.

The use of the SAML SSO Browser Profile extends the OGC Web Services interfaces in a standard conform way using the "vendor specific parameter" option. However, it still lacks interoperability. This is, because the use of a vendor specific parameter shall be used for experimental use only. Therefore, the OGC has to provide guidance in how identity information is to be transported from the client to the service. This could be done through Best Practices Paper or by extending the OWS Common Specification.

Further information on the topic of exchanging identity information and establishing trust relationships between OGC Web Services can be obtained from the "Trusted Geo Services IPR" (see [8]).

You can find more information on the above topics and demos as well as well as a reference implementation of a Policy Decision Point for deriving authorization decisions based on GeoXACML Policies on the GeoXACML homepage at www.geoxacml.org (see [9]).

As a final remark the author likes to apologize that not all concepts and topics could be discussed in the relevant and appropriate detail. In case you have any more questions, please do not hesitate and contact the author at
andreas.matheus(at)unibw.de.

BIBLIOGRAPHY

[1] **Open Geospatial Consortium, Inc.® (OGC)**
http://www.opengeospatial.org/

[2] **Geospatial eXtensible Access Control Markup Language (GeoXACML)**, Version 1.0, 20 February 2008, OGC
http://portal.opengeospatial.org/files/?artifact_id=25218

[3] **GeoXACML Implementation Specification - Extension A (GML2) Encoding (1.0)**, Version 1.0, 20 February 2008, OGC
http://portal.opengeospatial.org/files/?artifact_id=25219

[3] **GeoXACML Implementation Specification - Extension B (GML3) Encoding (1.0)**, Version 1.0, 20 February 2008, OGC
http://portal.opengeospatial.org/files/?artifact_id=25220

[4] **Organization for the Advancement of Structured Information Standards (OASIS)**
http://www.oasis-open.org

[5] **eXtensible Access Control Markup Language (XACML)**, Version 2.0, OASIS
http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

[6] **Security Assertion Markup Language (SAML),** Version 2.0, OASIS
http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip

[7] **Profiles for the OASIS Security Assertion Markup Language (SAML),** Version 2.0, OASIS
http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

[8] **Trusted Geo Services IPR**, OGC
http://portal.opengeospatial.org/files/?artifact_id=20859

[9] **GeoXACML homepage**, Andreas Matheus
http://www.geoxacml.org