OGC 20th Code Sprint – 25.-27.04.2023

Secure Dimensions contribution

**Statement of the Challenge**  
The OGC APIs are designed to work over HTTP(S). Securing the deployed APIs is not of concern to the OGC API Standards. However, the requirement to describe the API’s endpoint via OpenAPI introduces standard options to enable authentication. Knowing the user that accesses an API is an important aspect for many use cases. Also, user identification is often used to control access to the data, served by the API. But, no OGC API Standard provides guidance how to apply access control. This leads to an interoperability issue if access is denied, but there is no common (standardized) way to “tell the reason”. OGC’s GeoXACML 3.0 (currently still draft) Standard supports the interoperable exchange of access conditions as policies. GeoXACML 3.0 defines the geospatial extension to OASIS XACML 3.0 and thereby extends the ability to include Attribute Based Access Control also on geometries.

The challenge for the Codesprint is to develop a PEP (Policy Enforcement Point) that intercepts WFS 2.0 requests and modifies the request according to access decision returned by a GeoXACML 3.0 policy. The access control use case is the following: “WFS GetFeature requests shall exclude the NYC Central Park for feature type poly\_landmarks”.

**Purpose**  
Secure Dimensions implemented the GeoXACML 3.0 draft specification based on the Authzforce CE project which is an Open Source Community Edition project. The implementation is based on Java 11 and shall be made available as the Open Source reference implementation for GeoXACML 3.0. The implementation of a PEP during the Codesprint is important to test the GeoXACML 3.0 GeoPDP reference implementation by Secure Dimensions regarding request and response processing.

**Proposed Solution**  
With respect to implementation, a Geoserver deployment including the default data will be used as the API that is to be protected. From the Geoserver architecture, different possibilities exist where to place the PEP. For the Codesprint, the solution to deploy a Tomcat Filter is favored. This ensures that the Geoserver installation can almost remain unchanged. The configuration of the “web.xml” must just be adopted to include the “WFS Filter for GeoXACML 3.0”. Also, the filter’s JAR and dependencies must be added to “geoserver.war”.

The implementation of the WFS Filter itself will use the Authzforce XACML-SDK for Java that needs to be extended to allow the inclusion of Geometry attributes. The actual access condition to exclude all features of the Central Park should be modelled in a GeoXACML 3.0 Policy. This is possible by constructing an Obligation on Permit that returns the OGC Filter condition that the PEP needs to apply to the intercepted request.

**Achievement**  
During the Code Sprint, a demo was implemented that follows the XACML flow diagram.

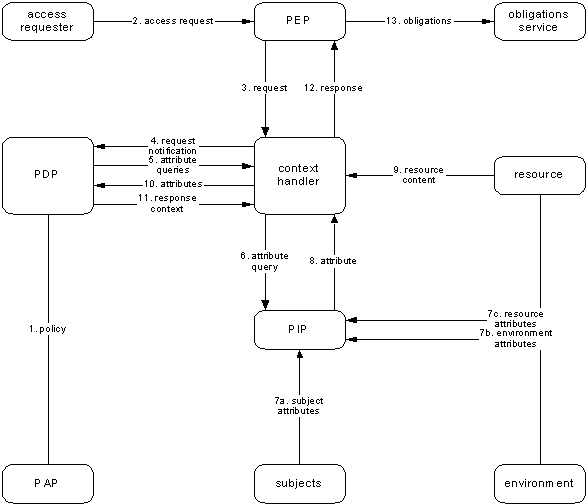


Figure: XACML Flow Diagram

A Geoserver 2.23 WAR-file deployment was used create the WFS 2.0 service endpoint.

The injection of the PEP was done by updating the “geoserver.war” file manually

* zip -u geserver.war WEB-INF/web.xml
* zip -u geoserver.war WEB-INF/lib/GeoXACMLFilter.jar

The web.xml file contains the deployment for the WFS Filter for GeoXACML 3.0.

|  |
| --- |
| <filter>  <filter-name>GeoXACMLFilter</filter-name>  <filter-class>de.securedimensions.geoxacml3.ows.GeoServerFilter</filter-class>  <init-param>  <param-name>pdpURL</param-name>  <param-value>https://ogc.demo.secure-dimensions.de/authzforce-ce</param-value>  </init-param>  <init-param>  <param-name>pdpDomain</param-name>  <param-value>A0bdIbmGEeWhFwcKrC9gSQ</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>Set Character Encoding</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <!-- Uncomment following filter to enable CORS  <filter-mapping>  <filter-name>cross-origin</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  -->  <filter-mapping>  <filter-name>GeoXACMLFilter</filter-name>  <url-pattern>/wfs</url-pattern>  </filter-mapping> |

Listing: Geoserver web.xml initializing the WFS Filter for GeoXACML 3.0 on path /wfs

During the Codesprint one bug concerning the processing of XML attributes was fixed.

**Demonstration**  
During the Code Sprint, the focus was to demonstrate the ability to rewrite WFS requests to reflect geospatial access control decisions. In order to achieve that, a simple GeoXACML 3.0 policy was crafted using ALFA for Visual Studio Code (ALFA plugin).

|  |
| --- |
| namespace ogc\_codesprint {  import GeoXACML.\*  import Attributes.\*  obligation Filter = "urn:secd:wfs:filter"  policyset Manhattan = "root" {  target clause service == "WFS"  apply permitOverrides  policy POLY\_LANDMARKS = "poly\_landmarks" {  target clause feature\_type == "tiger:poly\_landmarks"  apply firstApplicable  rule CENTRAL\_PARK {  target clause request == "GetFeature"  permit  condition(geometryOneAndOnly(bbox) >< "POLYGON((40.767774 -73.981464, 40.768268 -73.981396, 40.768483 -73.981634, 40.769272 -73.981131, 40.76983 -73.980652, 40.770488 -73.980215, 40.771096 -73.9798, 40.771753 -73.979298, 40.77241 -73.978862, 40.773018 -73.978447, 40.773708 -73.977923, 40.7743 -73.977465, 40.774859 -73.977072, 40.775565 -73.97657, 40.776288 -73.97609, 40.776947 -73.975588, 40.777554 -73.97513, 40.778244 -73.974671, 40.778852 -73.974234, 40.779427 -73.973776, 40.782105 -73.971899, 40.782795 -73.971375, 40.78342 -73.970917, 40.783995 -73.970437, 40.784685 -73.969956, 40.785293 -73.969498, 40.785983 -73.969018, 40.786624 -73.968537, 40.787264 -73.968036, 40.787922 -73.96762, 40.78848 -73.967119, 40.789105 -73.966704, 40.789713 -73.966245, 40.790353 -73.965787, 40.79106 -73.965241, 40.791684 -73.964804, 40.792358 -73.964281, 40.794247 -73.962905, 40.794905 -73.962403, 40.795545 -73.961966, 40.796153 -73.961529, 40.796761 -73.961049, 40.797418 -73.960612, 40.798125 -73.960109, 40.798782 -73.959607, 40.799374 -73.959149, 40.800047 -73.95869, 40.800425 -73.958428, 40.800507 -73.958124, 40.800588 -73.957885, 40.799509 -73.955312, 40.798298 -73.95248, 40.797003 -73.94954, 40.79669 -73.94952, 40.796329 -73.949761, 40.795705 -73.950241, 40.795031 -73.950744, 40.794374 -73.951159, 40.793684 -73.95177, 40.79306 -73.952142, 40.792419 -73.9526, 40.791729 -73.953103, 40.791154 -73.953496, 40.790414 -73.954042, 40.789199 -73.954937, 40.788624 -73.955352, 40.78795 -73.955854, 40.78726 -73.956335, 40.786669 -73.956815, 40.786028 -73.957339, 40.78542 -73.957732, 40.784796 -73.958212, 40.784188 -73.958627, 40.783514 -73.959086, 40.782873 -73.959588, 40.782233 -73.96009, 40.781625 -73.960548, 40.780852 -73.961029, 40.780294 -73.961466, 40.779587 -73.961946, 40.779012 -73.962383, 40.778388 -73.962863, 40.777747 -73.963343, 40.777106 -73.963845, 40.776334 -73.964391, 40.775726 -73.964871, 40.77502 -73.965438, 40.774494 -73.965939, 40.773771 -73.966398, 40.773196 -73.966856, 40.772523 -73.967315, 40.7718 -73.967817, 40.771225 -73.968253, 40.770585 -73.96869, 40.769992 -73.969148, 40.769368 -73.969607, 40.76871 -73.970065, 40.768135 -73.970501, 40.767511 -73.970981, 40.766837 -73.971397, 40.766213 -73.971898, 40.765605 -73.972313, 40.764981 -73.972793, 40.764389 -73.973251, 40.764621 -73.973791, 40.765651 -73.976428, 40.766812 -73.97926, 40.767575 -73.981008, 40.767774 -73.981464))":geometry)  on permit {  obligation Filter {  Attributes.Filter.operation = "Disjoint"  Attributes.Filter.geometry = "POLYGON((40.767774 -73.981464, 40.768268 -73.981396, 40.768483 -73.981634, 40.769272 -73.981131, 40.76983 -73.980652, 40.770488 -73.980215, 40.771096 -73.9798, 40.771753 -73.979298, 40.77241 -73.978862, 40.773018 -73.978447, 40.773708 -73.977923, 40.7743 -73.977465, 40.774859 -73.977072, 40.775565 -73.97657, 40.776288 -73.97609, 40.776947 -73.975588, 40.777554 -73.97513, 40.778244 -73.974671, 40.778852 -73.974234, 40.779427 -73.973776, 40.782105 -73.971899, 40.782795 -73.971375, 40.78342 -73.970917, 40.783995 -73.970437, 40.784685 -73.969956, 40.785293 -73.969498, 40.785983 -73.969018, 40.786624 -73.968537, 40.787264 -73.968036, 40.787922 -73.96762, 40.78848 -73.967119, 40.789105 -73.966704, 40.789713 -73.966245, 40.790353 -73.965787, 40.79106 -73.965241, 40.791684 -73.964804, 40.792358 -73.964281, 40.794247 -73.962905, 40.794905 -73.962403, 40.795545 -73.961966, 40.796153 -73.961529, 40.796761 -73.961049, 40.797418 -73.960612, 40.798125 -73.960109, 40.798782 -73.959607, 40.799374 -73.959149, 40.800047 -73.95869, 40.800425 -73.958428, 40.800507 -73.958124, 40.800588 -73.957885, 40.799509 -73.955312, 40.798298 -73.95248, 40.797003 -73.94954, 40.79669 -73.94952, 40.796329 -73.949761, 40.795705 -73.950241, 40.795031 -73.950744, 40.794374 -73.951159, 40.793684 -73.95177, 40.79306 -73.952142, 40.792419 -73.9526, 40.791729 -73.953103, 40.791154 -73.953496, 40.790414 -73.954042, 40.789199 -73.954937, 40.788624 -73.955352, 40.78795 -73.955854, 40.78726 -73.956335, 40.786669 -73.956815, 40.786028 -73.957339, 40.78542 -73.957732, 40.784796 -73.958212, 40.784188 -73.958627, 40.783514 -73.959086, 40.782873 -73.959588, 40.782233 -73.96009, 40.781625 -73.960548, 40.780852 -73.961029, 40.780294 -73.961466, 40.779587 -73.961946, 40.779012 -73.962383, 40.778388 -73.962863, 40.777747 -73.963343, 40.777106 -73.963845, 40.776334 -73.964391, 40.775726 -73.964871, 40.77502 -73.965438, 40.774494 -73.965939, 40.773771 -73.966398, 40.773196 -73.966856, 40.772523 -73.967315, 40.7718 -73.967817, 40.771225 -73.968253, 40.770585 -73.96869, 40.769992 -73.969148, 40.769368 -73.969607, 40.76871 -73.970065, 40.768135 -73.970501, 40.767511 -73.970981, 40.766837 -73.971397, 40.766213 -73.971898, 40.765605 -73.972313, 40.764981 -73.972793, 40.764389 -73.973251, 40.764621 -73.973791, 40.765651 -73.976428, 40.766812 -73.97926, 40.767575 -73.981008, 40.767774 -73.981464))":geometry  }  }  }  rule ALL\_PERMIT {  permit  }  }  }  } |

Listing: GeoXACML 3.0 PolicySet described in ALFA

The GeoXACML 3.0 policy is structured in a simple way:

* PolicySet (Manhattan) matches “service == WFS”
* Policy (POLY\_LANDMARKS) matches “typesNames == tiger:poly\_landmarks”
* Rule (CENTRAL\_PARK) matches “request == GetFeature”
* The Rule Condition contains is geospatial “BBOX Intersects Polygon(…)”

**Any request that matches the condition results in the decision “Permit” with the Obligation “urn:secd:filter”. As specified in the XACML 3.0 specification, a PEP must enforce the decision including all obligations. The processing of this filter obligation provides the missing information to construct the WFS Filter (disjoint Central Park).**

**The result of this processing can be visualized with QGIS (WFS Layer):**

****

**Figure: Left: Feature type “poly\_landmarks” without PEP -> Central Park feature(s) are included; Right: Feature type “poly\_landmarks” with PEP -> Central Park feature(s) are excluded!**

**The implementation of the Filter obtains the information from the HTTP request:**

|  |
| --- |
| SubjectCategory subjectCat = new SubjectCategory(); ResourceCategory resourceCat = new ResourceCategory(); ActionCategory actionCategory = new ActionCategory(); EnvironmentCategory environmentCategory = new EnvironmentCategory();  AttributeValueType serviceType = new AttributeValueType(Arrays.*asList*(httpRequest.getParameter("SERVICE")), XACMLDatatypeId.*STRING*.value(), null); Attribute service = new Attribute(Arrays.*asList*(serviceType),"urn:ogc:ows:service", "", false); resourceCat.addAttribute(service);  **...** |

**Using the XACML-SDK for Java from Authzforce, the response from the PDP can be obtained in a few lines of code:**

|  |
| --- |
| Request xacmlRequest = Utils.*createXacmlRequest*(Arrays.*asList*(subjectCat), Arrays.*asList*(resourceCat), Arrays.*asList*(actionCategory), Arrays.*asList*(environmentCategory));  ResponsesFactory xacmlResponse = pdp.getAuthZ(subjectCat, resourceCat, actionCategory, environmentCategory); for (Response r : xacmlResponse.getResponses()) {  LOGGER.info("XACML Response: " + r.toString());  DecisionType decision = r.getDecision();  LOGGER.info("XACML Decision: " + decision.toString());  LOGGER.info("decision: " + decision.value());  for (Obligation obligation : r.getObligations().getObligations()) {  if (obligation.getObligationId().equalsIgnoreCase("urn:secd:wfs:filter")) {  for (AttributeAssignment aa : obligation.getAttributeAssignments()) {  if (aa.getAttributeId().equalsIgnoreCase("urn:secd:filter:geometry")) {  filterGeometry = aa.getContent().get(0).toString();}  if (aa.getAttributeId().equalsIgnoreCase("urn:secd:filter:operation")) {  filterOperation = aa.getContent().get(0).toString();  }  }   }  } } |

**Lessons Learned**   
The Tomcat Filter implementation was based on Java 11. The existing XACML-SDK for Java was available for Java 8. Due to deprecation of javax classes in Java 11, JAXB related functionality had to be updated. The use of GeoTools to create the Filter programmatically could not be achieved. The unresolved problem was that the XML encoder did not include the CRS into the GML part of the spatial filter. Examples and documentation found did only cover non-spatial examples or BBOX Filter. But for implementing the use case, a Disjoint filter with a GML3 geometry had to be constructed. After removing GeoTools completely, a simple string template was used: “<fes:OPERATION><fes:ValueReference>the\_geom</fes:ValueReference>GEOMETRY</fes: OPERATION >” where the GEOMETRY was constructed from the response by the PDP (Obligation attributes urn:secd:filter:geometry and urn:secd:filter:operation).

|  |
| --- |
| <?xml version='1.0' encoding='UTF-8'?><ns4:Response xmlns:ns6="http://authzforce.github.io/pap-dao-flat-file/xmlns/properties/3.6" xmlns:ns5="http://authzforce.github.io/core/xmlns/pdp/8" xmlns:ns4="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" xmlns:ns3="http://www.w3.org/2005/Atom" xmlns:ns2="http://authzforce.github.io/rest-api-model/xmlns/authz/5"><ns4:Result><ns4:Decision>Permit</ns4:Decision><ns4:Obligations><ns4:Obligation ObligationId="urn:secd:wfs:filter"><ns4:AttributeAssignment AttributeId="**urn:secd:filter:operation**" Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource" DataType="http://www.w3.org/2001/XMLSchema#string">**Disjoint**</ns4:AttributeAssignment><ns4:AttributeAssignment AttributeId="**urn:secd:filter:geometry**" Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource" DataType="urn:ogc:def:geoxacml:3.0:data-type:geometry">**POLYGON** ((40.767774 -73.981464, 40.768268 -73.981396, 40.768483 -73.981634, 40.769272 -73.981131, 40.76983 -73.980652, 40.770488 -73.980215, 40.771096 -73.9798, 40.771753 -73.979298, 40.77241 -73.978862, 40.773018 -73.978447, 40.773708 -73.977923, 40.7743 -73.977465, 40.774859 -73.977072, 40.775565 -73.97657, 40.776288 -73.97609, 40.776947 -73.975588, 40.777554 -73.97513, 40.778244 -73.974671, 40.778852 -73.974234, 40.779427 -73.973776, 40.782105 -73.971899, 40.782795 -73.971375, 40.78342 -73.970917, 40.783995 -73.970437, 40.784685 -73.969956, 40.785293 -73.969498, 40.785983 -73.969018, 40.786624 -73.968537, 40.787264 -73.968036, 40.787922 -73.96762, 40.78848 -73.967119, 40.789105 -73.966704, 40.789713 -73.966245, 40.790353 -73.965787, 40.79106 -73.965241, 40.791684 -73.964804, 40.792358 -73.964281, 40.794247 -73.962905, 40.794905 -73.962403, 40.795545 -73.961966, 40.796153 -73.961529, 40.796761 -73.961049, 40.797418 -73.960612, 40.798125 -73.960109, 40.798782 -73.959607, 40.799374 -73.959149, 40.800047 -73.95869, 40.800425 -73.958428, 40.800507 -73.958124, 40.800588 -73.957885, 40.799509 -73.955312, 40.798298 -73.95248, 40.797003 -73.94954, 40.79669 -73.94952, 40.796329 -73.949761, 40.795705 -73.950241, 40.795031 -73.950744, 40.794374 -73.951159, 40.793684 -73.95177, 40.79306 -73.952142, 40.792419 -73.9526, 40.791729 -73.953103, 40.791154 -73.953496, 40.790414 -73.954042, 40.789199 -73.954937, 40.788624 -73.955352, 40.78795 -73.955854, 40.78726 -73.956335, 40.786669 -73.956815, 40.786028 -73.957339, 40.78542 -73.957732, 40.784796 -73.958212, 40.784188 -73.958627, 40.783514 -73.959086, 40.782873 -73.959588, 40.782233 -73.96009, 40.781625 -73.960548, 40.780852 -73.961029, 40.780294 -73.961466, 40.779587 -73.961946, 40.779012 -73.962383, 40.778388 -73.962863, 40.777747 -73.963343, 40.777106 -73.963845, 40.776334 -73.964391, 40.775726 -73.964871, 40.77502 -73.965438, 40.774494 -73.965939, 40.773771 -73.966398, 40.773196 -73.966856, 40.772523 -73.967315, 40.7718 -73.967817, 40.771225 -73.968253, 40.770585 -73.96869, 40.769992 -73.969148, 40.769368 -73.969607, 40.76871 -73.970065, 40.768135 -73.970501, 40.767511 -73.970981, 40.766837 -73.971397, 40.766213 -73.971898, 40.765605 -73.972313, 40.764981 -73.972793, 40.764389 -73.973251, 40.764621 -73.973791, 40.765651 -73.976428, 40.766812 -73.97926, 40.767575 -73.981008, 40.767774 -73.981464))</ns4:AttributeAssignment></ns4:Obligation></ns4:Obligations> |

The rewritten Filter was then processed by Geoserver:

|  |
| --- |
| typeNames[0] = {http://www.census.gov}poly\_landmarks  srsName = urn:ogc:def:crs:EPSG::4326  filter = [[ the\_geom within POLYGON ((40.46203574999999 -74.35610985937501, 40.46203574999999 -74.103704953125, 40.674587249999995 -74.103704953125, 40.674587249999995 -74.35610985937501, 40.46203574999999 -74.35610985937501)) ] AND [ the\_geom disjoint POLYGON ((40.767774 -73.981464, 40.768268 -73.981396, 40.768483 -73.981634, 40.769272 -73.981131, 40.76983 -73.980652, 40.770488 -73.980215, 40.771096 -73.9798, 40.771753 -73.979298, 40.77241 -73.978862, 40.773018 -73.978447, 40.773708 -73.977923, 40.7743 -73.977465, 40.774859 -73.977072, 40.775565 -73.97657, 40.776288 -73.97609, 40.776947 -73.975588, 40.777554 -73.97513, 40.778244 -73.974671, 40.778852 -73.974234, 40.779427 -73.973776, 40.782105 -73.971899, 40.782795 -73.971375, 40.78342 -73.970917, 40.783995 -73.970437, 40.784685 -73.969956, 40.785293 -73.969498, 40.785983 -73.969018, 40.786624 -73.968537, 40.787264 -73.968036, 40.787922 -73.96762, 40.78848 -73.967119, 40.789105 -73.966704, 40.789713 -73.966245, 40.790353 -73.965787, 40.79106 -73.965241, 40.791684 -73.964804, 40.792358 -73.964281, 40.794247 -73.962905, 40.794905 -73.962403, 40.795545 -73.961966, 40.796153 -73.961529, 40.796761 -73.961049, 40.797418 -73.960612, 40.798125 -73.960109, 40.798782 -73.959607, 40.799374 -73.959149, 40.800047 -73.95869, 40.800425 -73.958428, 40.800507 -73.958124, 40.800588 -73.957885, 40.799509 -73.955312, 40.798298 -73.95248, 40.797003 -73.94954, 40.79669 -73.94952, 40.796329 -73.949761, 40.795705 -73.950241, 40.795031 -73.950744, 40.794374 -73.951159, 40.793684 -73.95177, 40.79306 -73.952142, 40.792419 -73.9526, 40.791729 -73.953103, 40.791154 -73.953496, 40.790414 -73.954042, 40.789199 -73.954937, 40.788624 -73.955352, 40.78795 -73.955854, 40.78726 -73.956335, 40.786669 -73.956815, 40.786028 -73.957339, 40.78542 -73.957732, 40.784796 -73.958212, 40.784188 -73.958627, 40.783514 -73.959086, 40.782873 -73.959588, 40.782233 -73.96009, 40.781625 -73.960548, 40.780852 -73.961029, 40.780294 -73.961466, 40.779587 -73.961946, 40.779012 -73.962383, 40.778388 -73.962863, 40.777747 -73.963343, 40.777106 -73.963845, 40.776334 -73.964391, 40.775726 -73.964871, 40.77502 -73.965438, 40.774494 -73.965939, 40.773771 -73.966398, 40.773196 -73.966856, 40.772523 -73.967315, 40.7718 -73.967817, 40.771225 -73.968253, 40.770585 -73.96869, 40.769992 -73.969148, 40.769368 -73.969607, 40.76871 -73.970065, 40.768135 -73.970501, 40.767511 -73.970981, 40.766837 -73.971397, 40.766213 -73.971898, 40.765605 -73.972313, 40.764981 -73.972793, 40.764389 -73.973251, 40.764621 -73.973791, 40.765651 -73.976428, 40.766812 -73.97926, 40.767575 -73.981008, 40.767774 -73.981464)) ]] |

**Follow-Up**  
The OGC API-Features was not used as it is not yet possible to pass a Filter parameter. Once the Filter capability is standardized in OGC API-Features Part 3, the implementation for this demonstration could be adopted accordingly. The implementation would need to be adopted to obtain relevant information such as the feature-type, the request BBOX, etc. according to OGC API-Features.

**Conclusion**   
This Codesprint activity demonstrated the basics for enabling powerful Attribute Based Access Control including geospatial conditions. Regarding a production solution, the request rewriting to inject (not only) geospatial conditions via the OGC Filter parameter can be considered high-performance: little information is required to create a request in the PEP and send it to the GeoPDP. Likewise, the response from the GeoPDP to the PEP includes a decision and zero to multiple obligations that control the construction of the Filter. The deployment of the PDP is missing critical, but because this backend service is stateless, scaling is “straight forward” in cloud infrastructures.

A successful demonstration was given during the last day brief back. The QGIS application was used to connect to the GeoServer WFS using the regular OGC WFS 2.0 service. Invisible to the client, the PEP modified the WFS request sent by QGIS based on the access rights exptessed in the GeoXACML policy.