Contextualisation of ABAC Attributes through a Generic XACML Functionality Extension Mechanism

Brecht Claerhout[1], Kristof De Schepper[1], David Pérez del Rey[2] and Anca Bucur[3]
[1]*Custodix, Kortrijksesteenweg 214, 9830 Sint-Martens-Latem, Belgium*
[2]*Grupo de Informática Biomédica, Dept. Inteligencia Artificial, Facultad de Informática*
Universidad Politécnica de Madrid, Campus de Montegancedo S/N, 28660 Boadilla del Monte, Madrid, Spain
[3]*Phillips Research, Healthcare Information Management, High Tech Campus 34, 5656 AE Eindhoven, The Netherlands*

Keywords: Contextual Attributes, Contextual Roles, ABAC, XACML, Context Aware Policies, Extension Mechanism, Integrate Project, Breast Cancer.

Abstract: Authorisation solutions that exist today offer a broad range of functionality for defining complex access control policies. A common requirement that is not covered by these solutions is dynamically instantiated contexts in collaborative environments. This requirement is one of the research topics of the EU funded INTEGRATE project. This paper will focus on the solution proposed for the INTEGRATE project which is XACML based. The approach taken to make XACML context aware, is to enrich the XACML specification using a contextual extension through a generic mechanism, without changing the XACML language itself. This contextual extension operates on the XACML requests with ultimate goal to simplify the management of context aware policies.

1 INTRODUCTION

Several solutions have been proposed for defining and enforcing complex access control (AC) policies, such as for example XACML (OASIS, 2005), PERMIS (Chadwick et al., 2008), PONDER (Damianou et al., 2001), Cassandra (Becker and Sewell, 2004), etc. Although these solutions offer a broad range of functionality; still they logically cannot cover all specific demands of specialised application domains.

A common requirement of access management in collaborative environments is to be able to define default AC policies for dynamically instantiated contexts. A context instance defines the environment in which AC requests should be evaluated, e.g. a specific organisation in a wider cross-organisation collaboration. In some cases, one desires that instances of a context are governed by a set of "default" AC policies (rather than have their own specific policies). Typically this happens when context instances represent dynamic collaborations (e.g. Virtual Organisations (VOs) (Foster et al., 2001)).

In this paper, usage of contexts and context instances is further illustrated in the domain of Role Based Access Control (RBAC). In RBAC access decisions are made based on the role (e.g. investigator, administrator, etc.) the user has. The permissions to perform certain operations are assigned to specific roles. Usually it suffices to define roles for a user in a global context, however in some situations it is required to introduce roles that are specific to a given context (called contextual roles). The policies linked to these contextual roles are the same for each context but their scope is limited to each instance of the context. It is clear that when writing policies for contextual roles one does not want to define a policy for each role in each instance of a context.

2 BACKGROUND

The work described in this paper is the result of research within the INTEGRATE project (INTEGRATE, 2012). INTEGRATE is part of the FP7 framework funded by the European Commission and aims to develop innovative infrastructures to enable data and knowledge sharing and to foster large-scale collaboration in biomedical

Claerhout B., De Schepper K., Pérez del Rey D. and Bucur A..
Contextualisation of ABAC Attributes through a Generic XACML Functionality Extension Mechanism.
DOI: 10.5220/0004224700520057
In *Proceedings of the International Conference on Health Informatics* (HEALTHINF-2013), pages 52-57
ISBN: 978-989-8565-37-2

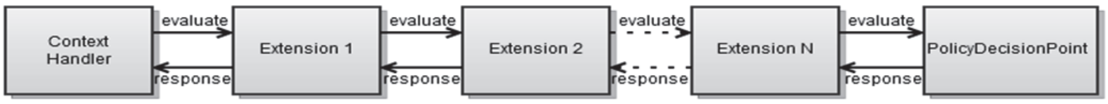


Figure 1: Pluggable extension handler pipeline.

research. After a thoroughly evaluation of the access control requirements in INTEGRATE it was decided to use XACML as authorisation solution. In this paper the proposed solution therefore focuses on XACML.

2.1 XACML

Attribute-Based Access Control (ABAC) presents an access control model inherently capable of meeting many of the “modern” access control demands (e.g. data dependent access policies, environment dependent policies, etc.). In ABAC, attributes that are associated with a user, action or resource serve as inputs to the decision of whether a given user may access a given resource in a particular way.

The eXtensible Access Control Markup Language (XACML) (OASIS, 2005) implements ABAC. It is a XML based declarative access control policy language defining both a policy, decision request and decision response language. XACML contains several profiles for supporting RBAC, multiple resources, hierarchical resource, etc. Although it offers a wide-range functional solution for access control, it lacks support for the problem of contextualisation.

2.2 Solution through Management Tools

A straightforward solution for contextualisation in XACML would be shifting the responsibilities to the policy authoring tools. This basically means that for each context instance separate policy files would need to be generated, e.g. from templates describing the default context policies. Although this approach seems easy to implement at first sight, it suffers from the inherent issues associated with all “auto-generating” solutions (be it for configurations or source code). Every change requires many policies to be rewritten (regenerated) and possibly redistributed. Furthermore, synchronisation becomes a big issue when one wants to allow exceptions in auto-generated policies (“manual” additions). For this reasons, this solution is not the most favourable for large scale environments such as INTEGRATE.

3 CONTEXTUAL ATTRIBUTES

3.1 Extending XACML Functionality

The main approach taken to enrich the XACML functionality with contextual attributes is not to make changes in the core XACML specification, such that no modifications are needed to the standard XACML access control decision mechanism (meaning standard XACML Policy Decision Points (PDP) can be used in implementations). Instead, the XACML requests are enhanced using a generic extension mechanism, in a way that extensions to the XACML specification can easily be added or be removed without needing to touch the core of the policy language and XACML compliant decision engines (see Figure 1). Extension handlers are placed in a pipeline between the context handler (strictly speaking the extensions are part of the context handler) and the PDP. An extension handler gets as input an XACML request from another extension handler or the context handler; subsequently transforms this request; and finally feeds the transformed request to the next extension or the PDP engine (last stage in the pipeline). This mechanism has previously been successfully applied for automatically translating attributes between different security contexts and semantically expanding them (Ciuciu et al. 2011).

3.2 Contextualisation Extension

The contextualisation extension handler proposed, splits up an incoming XACML request (containing possible multiple contexts and context instances in the attributes) in multiple single-context requests. More specifically, for each different context instance that is included in the original XACML request, a separate new context-specific XACML request is generated (see Figure 2). These context-specific requests are sent separately to the PDP engine for evaluation. By splitting up the requests, the PDP engine can provide an access decision for each context instance using manageable context-specific policies. The XACML responses containing the context-specific access decisions are sent back to the contextualisation extension handler where a global

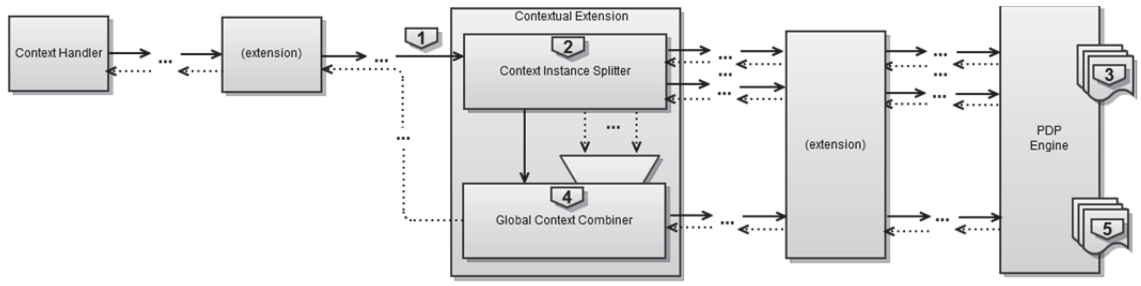


Figure 2: Contextual extension flow.

(no context specific) XACML request is generated. This request is then sent back to the PDP engine which evaluates the request, making use of defined global policies. The access decision is finally returned to the extension handler and is passed back to the access control requestor as final outcome of the original request.

3.2.1 Incoming XACML Request

To contextualise attributes of a user and mark the resources he/she wishes to access that are context-specific, a special attribute syntax is defined in the incoming XACML request that is recognizable by the contextual extension handler.

The example below shows how a contextual role attribute of a user is included in a request as an XACML subject attribute with attribute id “role” and a special formatted attribute string value containing the role, the context and the context instance:

```
<att id="role">
[context role]@[context]:[context instance]
</att>
```

For including the global context roles of a user in the request, this special formatting is not needed, because these roles are relevant for each of the contexts. In this case, the attribute value is only the role itself.

To illustrate the inclusion of contextual roles in more detail, an example is given in the domain of trial screening. A user called “John doe” can have different roles within different trials, e.g. a user can have the role of “clinical staff” over all trials (global context), the role of “investigator” in his own trial A (context trial A) and the role of “principal investigator” in another trial B (context trial B). These three roles of John are provided as subject attributes, as shown in Example 1.

```
<Request>
  <Subject>
    <att id="name">John Doe</att>
    <att id="role">investigator@trial:A</att>
    <att id="role">principal investigator@trial:B</att>
    <att id="role">clinical staff</att>
  </Subject>
  ...
</Request>
```

Example 1: Context roles of a user in an incoming XACML request (pseudo-code).

To indicate that a resource is context-specific, an extra predefined attribute is included in the corresponding XACML resource object with the attribute id “context" and a special formatted attribute string value containing a reference to the context and context instance:

```
<att id="context">
[context]:[context instance]
</att>
```

Note that a resource can belong to more than one instance of the same context. In Example 2, the user John Doe wants to access three resources named EHR001 in the trial context of trial A and trial B (multiple context instances), EHR002 in the trial context of trial B and EHR003 in the global context.

```
...
<Resource>
<att id="resID">EHR001</att>
<att id="context">trial:A</att>
<att id="context">trial:B</att>
<att id="type">crf</att>
</Resource>
<Resource>
<att id="resID">EHR002</att>
<att id="context">trial:B</att>
<att id="type">adm</att>
</Resource>
<Resource>
<att id="resID">EHR003</att>
<att id="type">doc</att>
</Resource>
```

Example 2: Context(s) of resources in an incoming XACML request (pseudo-code).

3.2.2 Context-specific Requests

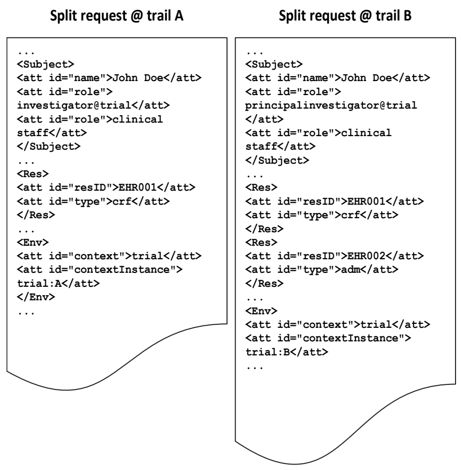
When a new XACML request enters the contextualisation extension handler, an inventory is made of all context and context instance included in this request. For each different context instance found, a new context-specific XACML request is generated.

For the context-specific role attributes in the example, this means that the role attribute of the original request is copied to the corresponding context-specific request. During this copying the context instance part is stripped from the role attribute value (this enables the generation of context-specific policies, see further). Non context-specific attributes are copied to each of the requests (so that they are available during contextual policy evaluation in each of the context instances).

Each context-specific resource object and the underlying resource attributes defined in the incoming request are copied to one or more of the corresponding context-specific requests, depending on the context instance(s) that is included in the context attribute value(s) of this resource (a resource can have more than one context/context instance). The context resource attribute(s) itself is omitted during the copying, because it is not relevant for evaluation. Non context-specific resources are not copied to a request, these will be used later in the global request (see further).

Finally in each generated context-specific request the corresponding context and context instance are added as environment attributes. This allows easy definition of policies concerning certain contexts or context instances. Other environment and action attributes in the original request are again copied to all context-specific requests.

Example 3 gives the result of splitting the incoming request attributes of the given above examples.


Split request @ trail A
Split request @ trail B

```
...
<Subject>
<att id="name">John Doe</att>
<att id="role">
investigator@trial</att>
<att id="role">clinical
staff</att>
</Subject>
...
<Res>
<att id="resID">EHR001</att>
<att id="type">crf</att>
</Res>
...
<Env>
<att id="context">trial</att>
<att id="contextInstance">
trial:A</att>
</Env>
...
```

```
...
<Subject>
<att id="name">John Doe</att>
<att id="role">
principalinvestigator@trial
</att>
<att id="role">clinical
staff</att>
</Subject>
...
<Res>
<att id="resID">EHR001</att>
<att id="type">crf</att>
</Res>
<Res>
<att id="resID">EHR002</att>
<att id="type">adm</att>
</Res>
...
<Env>
<att id="context">trial</att>
<att id="contextInstance">
trial:B</att>
...
```

Example 3: Splitted context-specific XACML requests (pseudo-code).

3.2.3 Context-specific Policies

After the request is split in multiple context-specific requests, these requests are sent to the PDP engine for evaluation. Because of this split up, the management and creation of context-aware policies is simplified. Generating policies for a context and/or context instance can be easily done in various ways, e.g. through the XACML "target" specification. Example 4 shows how to target a trial context (targeting all trial instances).
Example 5 gives the XACML code that needs to be specified to write a policy that targets a context instance trial A.

```
<Target>
  <Environments><Environment>
    <EnvMatch MatchId="string-equal">
      <AttValue>trial</AttValue>
      <EnvAttDesignator AttId="context"/>
    </EnvMatch>
  </Environment></Environments>
</Target>
```

Example 4: Context target (all instances).

```
<Target>
  <Environments><Environment>
    <EnvMatch MatchId="string-equal">
      <AttValue>trial:A</AttValue>
      <EnvAttDesignator AttId=
               "contextinstance"/>
    </EnvMatch>
  </Environment></Environments>
</Target>
```

Example 5: Context instance target.

Example 6 illustrates how a policy (or rule) could be written dealing with the access rights tied to a specific contextual attribute across instances (the role from the example in this case). Note that the attribute annotation "@[context]" (in the example, the "trail" context) facilitates policy authoring and review. It indicates that this is a contextual attribute and will thus be evaluated on a "per instance basis".

The examples illustrate that the presented mechanism for enhancing XACML with contextual attributes allows policies to be written in structured and manageable way.

```
<Target>
  <Subjects><Subject>
    <SubjectMatch MatchId="string-equal">
      <AttValue>
      principalinvestigator@trial
      </AttValue>
      <SubjAttDesignator AttId="role"/>
    </SubjectMatch>
  </Subject></Subjects>
</Target>
```

Example 6: Contextual role rule.

3.2.4 Global Request

The access control decisions for the different context-specific requests are sent back to the contextual extension handler. In the handler a new XACML request (global request) is generated based on the original access control request. This global request additionally contains the contextual access decision results for each resource.

To include the access results of the context-specific resources in this global request, one or more resource attributes "contextResult" are added for each resource. The attribute value has a special formatted string:

```
<att id="contextResult">
[Resource AC Decision]@[context]
</att>
```

Example 7 gives the generated request for the above example. This request is sent again to the PDP for evaluation.

This second evaluation, which includes the "contextResult" responses, allows "global" policies to be written that rely on the result(s) of the context-specific resources or override them (i.e. there is total freedom in combination algorithm). Example 8 shows how a rule could incorporate such a contextual result.

```
...
<Subject>
  <att id="name">John Doe</att>
  <att id="role">clinical staff</att>
</Subject>
...
<Res>
  <att id="resID">EHR001</att>
  <att id="contextResult">permit@trial</att>
  <att id="contextResult">deny@trial</att>
  <att id="type">crf</att>
</Res>
<Res>
  <att id="resID">EHR002</att>
  <att id="contextResult">permit@trial</att>
  <att id="type">adm</att>
</Res>
<Res>
  <att id="resID">EHR003</att>
  <att id="type">doc</att>
</Res>
...
</Env>
...
```

Example 7: Global request.

```
...
<Rule Effect="Permit">
    <Condition>
        <Apply FunctionId="any-of">
            <Apply FunctionId="string-equal"/>
            <AttValue>permit@trial</AttValue>
            <SubjAttDesignator
               AttId="contextResult"/>
        </Apply>
    </Condition>
</Rule>
...
```

Example 8: Global policy.

Note that in the special case where a resource belongs to more than one context instance of the same context, both a "permit" and "deny" for the same context could be present. Although this seems contradictory at first, it allows decision combination to be specified in the XACML policies and thus gives total freedom to policy writers.

Finally, the resulting XACML response, containing the access result for each resource is sent back to the contextualisation extension handler. The handler in its turn will return this response as outcome to "calling" handler.

4 CONCLUSIONS

In this paper the problem of contextualisation of ABAC attributes through a generic XACML functionality extension mechanism is described. Today's prominent access control solutions do not provide sufficient support for the contextualization of attributes. The solution that was proposed, is based on ABAC/XACML because the relevance in the INTEGRATE project.

The approach that was taken to enrich XACML with contextual attributes is not to change the specification of XACML itself but provide a generic flexible extension mechanism compatible with the standard XACML core specification.

The contextual extension handler described, splits up a request containing different contexts to multiple context-specific requests. After evaluation of these requests, a global request is created which in his turn is evaluated by the PDP. The advantage of this extension is that policies can be written specific for one context without making these policies very complex.

ACKNOWLEDGEMENTS

This project/study is partially funded by the European Commission under the 7th Framework Programme (FP7-ICT-2009-6-270253).

REFERENCES

OASIS, 2005, 'XACML: eXtensible Access Control Markup Language', Version 2.0, Available from: <http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf>. [16 July 2012]

Chadwick, D, Zhao, G, Otenko, S, Laborde, R, Su, L, Anh Nguyen, T, 2008, 'PERMIS: a modular authorization infrastructure', *Concurrency and Computation: Practice and Experience*, vol. 20, no. 11, pp. 1341-1357

Damianou, N, Dulay, N, Lupu, E, Sloman, M, 2001, 'The Ponder Policy Specification Language', POLICY 2001 Proceedings of the *International Workshop on Policies for Distributed Systems and Networks*, pp. 18-38

Moritz, Y, Becker, PS, 2004, 'Cassandra: Distributed Access Control Policies with Tunable Expressiveness, Policies for Distributed Systems and Networks', POLICY 2004 *Proceedings of the Fifth IEEE International Workshop*, pp. 159 – 168

Ciuciu, I, Claerhout, B, Schilders, L, and Meersman, R, 2011, 'Ontology-Based Matching of Security Attributes for Personal Data Access in e-Health', *OTM'11*, vol. 2, pp. 605-616

INTEGRATE, 2012, 'INTEGRATE: Driving Excellence in Integrative Cancer Research', Available from: <http://www.fp7-integrate.eu/>. [16 July 2012]

Foster, I, Kesselman, C, Tuecke, S, 2001, 'The Anatomy of the Grid: Enabling Scalable Virtual Organizations', *International Journal of Supercomputer Applications*, vol. 15, no. 3, pp. 200-222