On XACML, role-based access control, and health grids

D. Power, M. Slaymaker, E. Politou and A. Simpson

Contents

Background

Within the UK, the National Health Service (NHS) comprises a number of hospital trusts, each of which is an independent legal entity. Each hospital trust is legally responsible for the data held at its sites: this data is released only with respect to the principles of the Caldicott Guardian, which include:

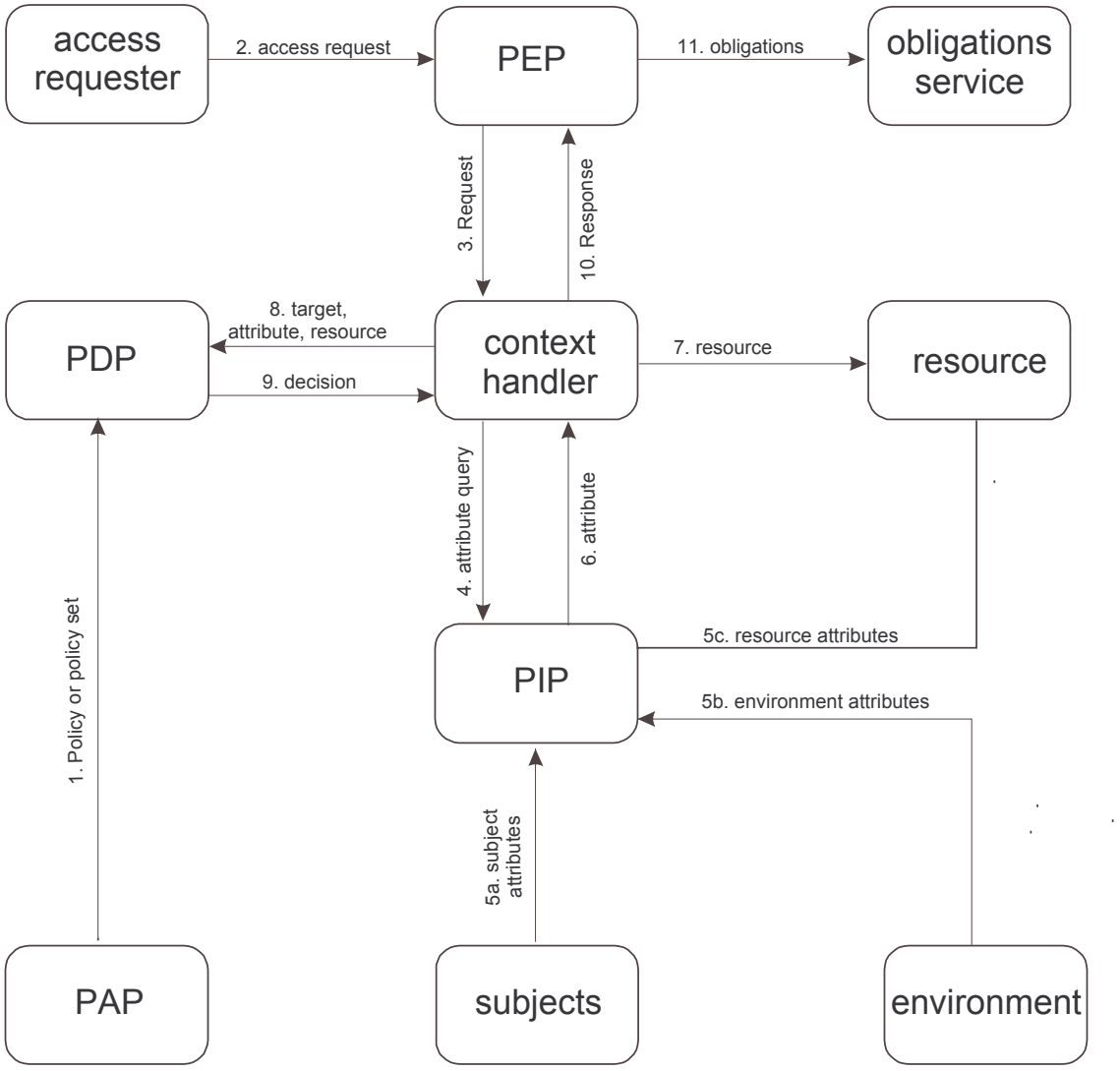
- Don't use patient-identifiable information unless it is absolutely necessary.
- Use the minimum necessary patient-identifiable data.
- Access to patient-identifiable information should be on a strict need to know basis.
- Everyone should be aware of their responsibilities.

XACML

XACML (eXtensible Access Control Markup Language) is an OASIS standard that allows one to describe in terms of XML two languages for a particular application.

The first language the policy language is used to describe general access control requirements. The policy language has standard extension points that allow one to define aspects such as new functions, data types, and logics to combine such entities.

The second language the request/response language allows one to construct a query to determine whether a particular action should be permitted. Every response contains an answer pertaining to whether the action should be permitted.



XACML Architecture

Role Based Access Control

Role-based access control has gained in popularity in recent years, with one of the principal reasons for this being that the approach offers the benefits of greatly simplified policies by exploiting the natural hierarchies of the workplace. Further, many vendors have started to incorporate role-based access control into their products.

The approach allows privileges to be associated with arbitrary roles, and, in turn, these roles can be associated with users. This offers the benefits of providing increased granularity and reducing administrative effort.

Given the legal and ethical requirements that are incumbent upon us in the health domain we may conclude that from an access control point of view the access given to a user will depend on many factors.

To be able to exploit the advantages of role-based access control without limiting all access decisions to be entirely based on a users role it is first necessary to be able to mix role-based decisions with decisions based on other factors.

As a generic and extensible language, XACML is fully capable of supporting such policies. However, as we shall see, the standard RBAC profile adds some undesirable limitations, which it has been necessary for us to remove.

XACML RBAC Profile

To address some of the problems associated with role-based access control in XACML there is a specific role-based access control profile.

Assuming that the roles of a subject have already been determined (or can be determined at the time the policy is evaluated), the profile suggests the formation of two special types of Policy Sets to describe the permissions of various roles. These are called the Role Policy Set (RPS) and the Permission Policy Set (PPS). A Role Policy Set is only applicable to a specific role. Each role has its own Role Policy Set. The actual permissions for a role are specified in a Permission Policy Set, which is referenced by the Role Policy Set. This extra level of indirection allows one role to possess all the permissions of another role by including a reference to the junior roles Policy Permission Set in the senior roles Policy Permission Set.

An Example

We will consider a health grid with just five users and four roles, with the roles being Doctor, Nurse, Administrator and Security Cleared. Some users have multiple roles as indicated in the sets below.

- User_1 -> Doctor
- User_2 -> Nurse
- User_3 -> Nurse, Admin
- User_4 -> Admin
- User_5 -> Doctor, Security

The roles have the following relationships.

- nurses are more junior than doctors; and
- administrators are more junior than doctors.

The permitted actions for each of the roles are summarised in the following rules.

- Doctors can prescribe drugs.
- Nurses can administer drugs.
- Administrators can modify patient records.
- You need to be both a Nurse and an Administrator to register a new patient.
- Security cleared personal are the only ones that can deal with secret patients.

The following should be noted:

- the first three rules grant permissions to a specific role;
- the fourth rule requires two roles for an action to be permitted; and
- the fifth rule restricts permissions given in the previous rules.

Implementation using the RBAC Profile

To define the policies needed for the first three rules is straightforward using the XACML RBAC profile.

The doctor Permission Policy Set would contain a reference to both the nurse Permission Policy Set and the administrator Permission Policy Set, and, as such, would only need to contain details of the prescribe action.

The administer action would be part of the nurse Permission Policy Set.

The modify action would be part of the administrator Permission Policy Set.

The fourth rule states that one needs to be both a nurse and an administrator to register a new patient.

A new Role Policy Set can be written that applies only to users who have both the role nurse and the role administrator.

However, this has the drawback that doctors who are senior to both nurses and administrators would not automatically gain the right to register new patients.

An additional reference needs to be added to the doctor Permission Policy Set so that it inherites the permissions of the combined nurse and administrator Permission Policy Set.

The fifth rule is more complicated as it restricts permissions that have already been given in previous rules.

In generic XACML policies it is possible to combine conflicting results using a range of different algorithms. In the RBAC profile the algorithm used is permit overrides, which allows permission to be granted if any of the Permission Policy Sets give permission to an action.

To add the fifth rule would require the use of a different algorithm such as deny overrides, which would only give permission for an action if there is at least one policy that permits the action and none of the other policies deny the action.

Using deny overrides would not be in keeping with the RBAC profile, but is a perfectly valid thing to do in a generic XACML policy set.

Possible Solutions

The problems that were found when trying to use the RBAC profile associated with our example were, in part, due to the desire to represent the hierarchy of roles by including a reference to one Permission Policy Set wihin another.

In addition, the decomposition of policies by role may not be the most natural way to structure a set of policies for a particular access control scenario.

What would be more desirable would be to have the ability to ask the question does the subject of the request have role X? either directly or through having a role that is senior to role X.

The simple solution to this problem is to define a subject attribute for each role. Such an attribute would be a Boolean attribute that represents the notion of having the permissions of that role. As this attribute respects the role hierarchy it will not be necessary to refer to one Permission Policy Set within another as the subject will appear to have all roles junior to those it possesses.

The problem with this simple solution is that it requires calculations to be performed outside of the PDP. This involves extending the functionality of the system making it specific to a single implementation; as such it cannot be thought of as a generic solution.

An Alternative Implementation

Built using the Sun Microsystems implementation of the XACML standard.

Added a new attribute type called string-pair type can be defined in the following way.

```
<AttributeValue DataType="string-pair">(Doctor,Nurse)</AttributeValue>
```

Added a new function, called is-role-function, which takes three arguments:

- a bag of string-pair, which contains the role hierarchy;
- a bag of string, which represents all of the subjects
- a string, which represents the role which is required.

The function returns true if the required role is junior or equal to any of the subjects roles and false otherwise.

The function can be used either using a local role hierarchy,

```
<Condition FunctionId="is-role-function">
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-pair-bag">
<AttributeValue DataType="tuple">Doctor,Nurse</AttributeValue>
<AttributeValue DataType="tuple">Doctor,Admin</AttributeValue>
</Apply>
<SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
AttributeId="roles" />
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Doctor</AttributeValue>
</Condition>
```

or one defined in the enviroment.

```
<Condition FunctionId="is-role-function">
<EnvironmentAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string-pair"
AttributeId="role-hierarchy" />
<SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
AttributeId="roles" />
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Doctor</AttributeValue>
</Condition>
```

Conclusions and Further Work

The RBAC profile for XACML aims to provide support for role-based policies while retaining the full expressive power of generic XACML. It is capable of handling role hierarchies using an inclusion mechanism which allows the permissions of one role to be inherited by another role.

With the aid of a simple example we have shown that use of the RBAC profile places undesirable constraints on the policies that can be constructed.

Two possible alternative mechanisms were proposed: the first assumes that the context handler knows how to evaluate role inheritance; the second extends the language to support a pair of strings data type and adds a custom function to handle role hierarchies.

Both of the proposed mechanisms either involve storing the role hierarchies outside of the policies or repeatedly stating definitions on every usage in every policy file in which they are used.

This is not desirable as the role hierarchies are very much part of the policy and repeating the definitions may lead to discrepancies. Using the alternative of allowing declarations of attributes as part of a policy, coupled with an extension to the policy reference mechanism that is currently available, will allow data members representing role hierarchies to be shared between policies and policy sets from a single file.

Other avenues for future work include extending the pair datatype to handle arbitrary tuples. This would be desirable for many contexts, not least of which is the representation of results from database queries.

Summary

- Background
- XACML
- Role Based Access Control
- An Example
- Implementation using the RBAC Profile
- Possible Solutions
- An Alternative Implementation
- Conclusions and Further Work