On XACML, role-based access control and health grids

David Power, Mark Slaymaker, Eugenia Politou and Andrew Simpson

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD

Abstract

In this paper we discuss the use of XACML policies to describe access control for use in health grids. We discuss the motivations for the use of role-based access control (RBAC) in a medical context. Using a simple motivating example we consider the use of the RBAC profile for XACML and discuss the limitations of the current profile. Finally, we discuss various solutions to providing fully flexible RBAC support in XACML, and consider their impact on the existing language.

1. Introduction

XACML[1] has established itself as a de facto standard for access control within grid environments, with the benefits offered being considerable. For example, in [1] the following benefits (amongst others) are offered.

- One standard access control policy language can replace dozens of application-specific languages.
- Developers save time and money because they don't have to invent new policy languages and write code to support them.
- Good tools for writing and managing XACML policies will be developed, since they can be used with many applications.
- XACML is flexible enough to accommodate most access control policy needs and extensible so that new requirements can be supported.
- One XACML policy can cover many resources, which helps avoid inconsistent policies on different resources.

The recent release of XACML version 2.0 has further cemented this position as a de facto standard.

In this paper we consider the use of XACML within health grid environments. In particular, we consider the use of XACML within the context of the development of a secure, interoperable web services infrastructure that is sympathetic to the legal and ethical requirements associated with healthcare delivery and research within the United Kingdom.

[1] www.oasisopen.org/committees/tc_home.php?wg_abbrev=xacml

2. Requirements for secure health grids

In [2], a vision for secure grid-enabled healthcare is presented. The paper details requirements for a secure infrastructure that supports healthcare delivery and research within the United Kingdom, with the requirements being derived from a number of projects that the authors have been charged with delivery for, including e-DiaMoND [3] and GIMI (Generic Infrastructure for Medical Informatics) [4].

The requirements for an access control model for the systems that we are developing can be simply stated: it should be flexible and fine-grained. This is entirely due to the fact that there is no way of stating *a priori* what access control policies will be implemented at each site: some requirements will be nationally (or internationally) mandated, while others will be due to local policies and requirements. The best that one can do in such circumstances is to offer a system that is sufficiently flexible to accommodate these different needs

The National Health Service comprises a number of independent legal entities know as *hospital trusts*. 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 the following [5].

- Every proposed use or transfer of patient-identifiable information within or from an organisation should be clearly defined and scrutinised.
- Don't use patient-identifiable information unless it is absolutely necessary.
- Where use of patient-identifiable information is considered to be essential, each individual item of information should

be justified with the aim of reducing identifiability.

- Access to patient-identifiable information should be on a strict need-to-know basis: only those individuals who need access to patient-identifiable information should have access to it, and they should only have access to the information items that they need to see.

As each trust retains the ownership of all data located at its site, coupled with the fact that each trust determines who can access its data (and under what circumstances), it can be concluded that the NHS constitutes a virtual organisation.

3. Our security architecture

Before considering role-based access control and XACML, it is worth providing a brief overview of the security architecture developed as a result of our experiences within e-DiaMoND and GIMI.

At each node the services are grouped into internally and externally facing services and the virtualisation of the data sources is assumed to take place at the service level. This architecture allows each node to retain full control of its data and to determine who can access it (and when), in accordance with the principles of the Caldicott Guardian. All user interactions with a site are made via the externally facing services, and it is only these externally facing services that are required to present a consistent interface. Figure 1 shows a representation of the architecture.

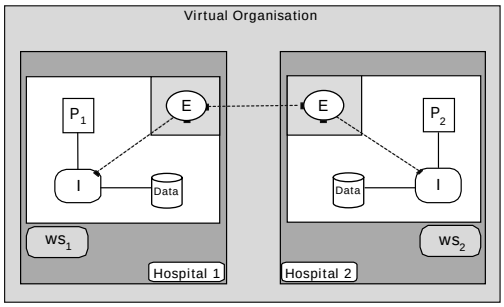

Figure 1.

4. Role-based access control

Role-based access control has gained in popularity in recent years. One of the principal reasons for this is that the approach offers the benefits of greatly simplified policies by exploiting the natural hierarchies of the workplace. At first sight such a system would be ideal for the medical domain with its rigid hierarchy of consultants, doctors, and nurses. However from an access control point of view the effective role of a person will depend on many factors including – but not limited to – the patient in question, the hospital that owns the data, the location of the user, and the previous history of changes made to the data.

5. Limitations of current technologies

In [6], the current authors described the use of the Distinguished Name (DN) from an X509 certificate to describe the role(s) of a user. This is relatively simple to refer to in XACML, which has a whole class of X500 functions that allow access to all or part of a DN. This, however, assumes that the roles of a user are static and do not vary between locations. This is not the case in our vision of a secure health grid: for example, there may be occasions when a doctor at a remote location has to access information (perhaps in an emergency) pertaining to one of her patients. In addition, we would also like to support the delegation of access rights, the details of which would need to be stored separately if the DN was used to store roles.

A more natural method of deciding what role a user has at a particular hospital is to use locally stored sets of users. Role hierarchies can be stored in a similar manner. This does not preclude any information being stored in the DN, but adds considerably to the flexibility of local hospitals to decide their own access control policies – which is, of course, consistent with the notion of data ownership that we attempt to support.

XACML uses bags (multi-sets) to store data; these can be converted to sets and support all the standard set operations, including set membership. However the standard method for including such data is not for it to be part of the policy but to be embedded into the request. This can be seen in the example request in the XACML specification [1] in which Julius Hibbert wishes to access the date of birth of Bart Simpson. The original data is assumed to be stored in an XML file which Julius has requested access to, but in the example the actual data from the XML file is embedded into the request!

6. The RBAC profile

To address some of the problems associated with role-based access control in XACML there

is a specific profile. However, the scope of this profile does not include answering the question "What set of roles does subject X have?": this is left to a Role Enablement Authority, the implementation of which is not specified.

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 PolicySets 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 role's Policy Permission Set in the senior role's Policy Permission Set.

7. An example

As a simple 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.

Here, we use a traditional RBAC interpretation of "junior". Doctors will have all the permissions of both nurses and administrators.

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

1. Doctors can prescribe drugs.
2. Nurses can administer drugs.
3. Administrators can modify patient records.
4. You need to be both a Nurse and an Administrator to register a new patient.
5. 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.

8. Possible solutions

To define the rules needed for the first three conditions is straightforward using the XACML RBAC profile. The Doctor PPS would contain a reference to both the Nurse PPS and the Administrator PPS, and as such would only need to contain details of the "prescribe drugs" action, as the administer drugs action would be part of the Nurse PPS and the modify patient record action would be covered by the Administrator PPS.

Rule four, which states that you need to be both a Nurse and an Administrator to register a new patient, introduces additional complications. A new Role Policy Set could be written which 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. For this to happen, an additional reference would need to be added to the Doctor PPS to inherit the permissions of the combined Nurse and Administrator PPS.

Rule five adds another complication as it restricts permissions that have already been given in previous rules. This does not present a problem in generic XACML policies as it is possible to combine conflicting results using a range of different algorithms. For instance, in the examples of the RBAC profile the algorithm used is permit-overrides which allows permission to be granted if any of the PPSs give permission to an action. To add rule five 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.

To implement rule five would involve creating a policy that relates to secret patients and will return permit if the subject has the role Security

Cleared and deny if the user does not have the role. This would not be in keeping with the RBAC profile but is a perfectly valid thing to do in a generic XACML policy set. By using the deny-overrides algorithm to combine this policy with the existing role policy sets already defined, rule five can be successfully implemented.

9. Determing a user's roles

As has been already mentioned it is not possible to have data stored as part of the policies. A sensible approach to the problem is to use attributes to store the data related to roles. In version 1.0 of the standard, all such attributes would be accessed by the context handler on the sole basis of the request. This puts the responsibility to request the correct data on the context handler, which would not be privy to the current state of a possibly dynamic policy. Fortunately in versions 1.1 and 2.0 of the standard, it is possible for the PDP to request the appropriate attributes based on the contents of the policy. This is, of course, a much more sensible state of affairs. However, it still leaves the problem of how the context handler should store and retrieve the attribute data. The exact method used is intended to be implementation dependent.

The problems that were found when trying to use the RBAC profile describe in the last section were in part due to the desire to represent the hierarchy of roles by including a reference to one PPS in 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 problem. What would be more desirable would to be able 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, which is 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 PPS in another as the subject will appear to have all roles junior to those it possesses. Indeed it is no longer necessary to use RPS/PPS pairs to represent the permissions of a role as the indirection no longer serves a purpose.

The problem with this simple solution is that it requires calculations to be performed outside of the PDP. This would involve extending the functionality of the system making it specific to a single implementation; as such it cannot be thought of as a generic solution. However, moving the calculation into the PDP is also problematic (as is discussed in the next section).

10. Extending XACML

XACML offers a number of different primitive data types such as String, Integer and Boolean. There are also bags of primitive data types. However, bags may not contain other bags, and – as such – there is no mechanism for describing more complex collections of data.

The only mechanism for inserting data values into a policy is by explicitly including them as arguments to functions. This increases the complexity of the most logical method of defining a role hierarchy as part of the policy as to do so would require the explicit redefinition of the role hierarchy every time it is used as an argument of a function. This problem could be avoided if static definitions were allowed in policies. If this were the case it would be possible to have a single definition of the role hierarchy that could be used as an argument to functions throughout a policy. It would also be possible through an "include-like" statement to use a single definition for a whole collection of policies: this would remove the possibility of duplication errors.

The only functions that can be used in a policy are those that have been pre-defined: there is no mechanism for defining new functions.

With such a limited amount of scope for defining non-trivial policies, the only mechanism left is to use the extensibility of the language by defining our own data types and functions.

What we would like to do is to use two attributes: one from the subject representing their roles and one from the environment representing the role hierarchy.

The roles of a subject can be easily represented as a bag of strings. The role hierarchy presents a bit more of a problem as it is effectively a bag of pairs. Even representing a single pair is problematic, as bags do not have any ordering. It would not be possible to tell the difference between a nurse being junior to a doctor, or a doctor being junior to a nurse.

One solution to the lack of pairs would be to use the number of times a value appears in a bag as representing its position. This obviously will not work if the two values are not distinct but that would not be a problem in terms of a role hierarchy. This solution will not get very far however as we would like a bag of pairs and bags are not permitted to contain bags.

So the first extension we would need would be the introduction of a new primitive type of pairs of strings. It should be noted that due to the way primitive types are referred to, it would not be possible to define a generic pair which could contain any two (possibly equal) data types.

The next problem is to add sufficient functions so that the following question can be answered. Does the set of the user's roles contain a role that is senior or equal to a specific role according to a specified role hierarchy. It would be possible to add this as a custom function, or to break down its functionality more and define a number of possibly more generally applicable functions.

11. Conclusions

XACML is emerging as a de facto standard for access control. It aims to be as generic as possible and to support a wide range of access control problem domains. It is also extensible allowing new problems to be solved which have not already been considered.

We have briefly considered the requirements for access control policies for health grids in general and for e-DiaMoND and GIMI in particular.

Role-based access control offers a means of simplifying access control policies by grouping together sets of people and considering them as one entity. It is useful in a health grid setting as the medical domain contains a number of natural hierarchies. However, we would still like to retain the ability to write policies that are more flexible and depend on the individual user and not just their role.

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.

The RBAC profile provides support for permissions that require multiple roles, but the mechanism fits poorly with inheritance mechanism used for role hierarchies.

The RBAC profile assumes that permissions are additive and that it is natural to write separate policies for each role. We have given a very simple example where this is not the case.

To add more flexible support for roles and role hierarchies to XACML it is necessary to extend the current language. Two mechanisms where proposed; either assume the context handler knows how to evaluate role inheritance, or extend the language to support a pair of strings data type and add some custom functions that can calculate the hierarchies.

Both of these mechanisms involve storing the role hierarchies outside of the policies. This is not desirable as the role hierarchies are very much part of the policy. An alternative would be to allow the declaration of static data members as part of a policy. Coupled with a simple inclusion mechanism so that these static data members can be shared between policies and policy sets it would be possible to define the role hierarchies in a single file.

12. References

1. Sun's XACML implementation. http://sunxacml.sourceforge.net

2. D. J. Power, M. A. Slaymaker, E. A. Politou, and A. C. Simpson. Towards secure Grid-enabled healthcare. Software Practice and Experience, 2005; 35:857-871

3. J. M. Brady, D. J. Gavaghan, A. C. Simpson, R. P. Highnam, and M. Mulet-Parada. "e-DiaMoND: A grid-enabled federated database of annotated mammograms". In Grid Computing: Making the Global Infrastructure a Reality. Wiley, 2002: 923-943.

4. A. C. Simpson, D. J. Gavaghan, D. J. Power, M. A. Slaymaker, S. Lloyd. and E. Politou. "GIMI: Generic Infrastructure for Medical Informatics". In Proceedings of Computer-Based Medical Systems, IEEE CS Press:564-566

5. "Medical ethics and law: confidentiality, data protection, Caldicott principles, computer use and patient records". www.addenbrookes.org.uk/advice/

medethlaw/confidential1.html

6. D. J. Power, M. A. Slaymaker, E. A. Politou, and A. C. Simpson. "A secure wrapper for OGSA-DAI utilising XACML". In Proceedings of EGC 2005, Springer-Verlag:485-494