# Managing the lifecycle of XACML delegation policies in federated environments

Manuel Sánchez, Óscar Cánovas, Gabriel López, Antonio F. Gómez-Skarmeta

**Abstract** This paper presents an infrastructure that enables the use of administrative delegation in an effective way, reducing the complexity in the policy management for some specific scenarios. This infrastructure is in charge of managing the policies of the system during its lifecycle, for example when they are created by the users or when they are collected to take an authorization decision. The proposal makes use of a robust and extensible language as XACML in order to express the authorization policies. However, as we will see, the management infrastructure has been designed in a way that facilitates the task of the different users involved, assuming that those users do not have to be security experts or XACML-aware.

## 1 Introduction and motivation

Nowadays we are experiencing the emergence of federated approaches to resource sharing, like eduroam [5]. In this approaches, trust links are established among different autonomous institutions in order to grant users in any of them access to shared resources with a single identity, stated by the institution the user belongs to. Moreover, this kind of agreements is facilitating mobility of users among institutions. In this scenarios, access control policies are used in order to manage the access of users to the services and resources offered by an institution. But due to the dynamic nature of this kind of scenarios, the management of that policies becomes more difficult. In this situations, *administrative delegation* [6] can help to reduce this complexity. It means that the system administrator can delegate in another person, the *delegate*, to make part of the work by managing a subset of the policies.

Manuel Sánchez, Gabriel López, Antonio F. Gómez-Skarmeta
Department of Information and Communications Engineering, University of Murcia, Spain
e-mail: {manuelsc, gabilm, skarmeta}@um.es

Óscar Cánovas
Department of Computer Engineering, University of Murcia, Spain e-mail: ocanovas@um.es

A scenario that shows this situation could be a meeting of an international project. In this meeting there are people from different institutions who belong to the same federation. During the meeting time, the host institution wants to provide to the members Internet access. Besides, some constraints such as the use of a specific Virtual LAN (VLAN) or a Quality of Service (QoS) profile must be enforced. In this case, administrative delegation can be used to assign the responsability of managing the access control properties to a user more closely related to the mentioned scenario. For example, the person who is organizing the meeting can be the delegate in this situation, because he knows all the necessary information such as the identity of the audience or the schedule of the meeting. In this way, the system administrator will create one policy to delegate the network access management to the meeting organizer. Then, the meeting organizer will create the specific policies to control the access to the network.

This scenario can help us to show the different issues involving the administrative delegation. For example, the delegate does not usually know anything about policy management, therefore it is necessary to help them to develop a correct access control policy. Moreover, we have additional policies in the system to specify the delegation. Thus, despite the work of the system administrator is reduced, the management of the policies is more complex from a global point of view. An infrastructure for managing the policies is necessary in order to help the system administrator and delegates to carry out their tasks. Thus, our main contribution in this paper is the definition of a complete infrastructure that provides the means of controlling how the policies are created, where they are stored or how the appropriate policies are collected and evaluated when it is necessary to take an authorization decision.

The rest of this paper is structured as follows. Section 2 shows the access control language used in this proposal and its extension to enable the use of administrative delegation. Finally, section 3 describes the proposed infrastructure.

## 2 Administrative delegation in XACML

XACML (eXtensible Access Control Markup Language) [3] is a standard XML-based access control language proposed by OASIS to represent access control policies in a standard way. This specification includes the definition of an access control policy language and a representation format to encode access control requests and responses. An XACML delegation profile [8] was produced to support administrative and dynamic delegation. The purpose of this profile is to specify how to express permissions about the right to issue policies and to verify issued policies against these permissions. Basically, it defines that an XACML policy may contain a *PolicyIssuer* element that describes the source of the policy. Then, the authority of the issuer needs to be verified in order to consider this policy as valid. The essence of the verification is that the issuer of the policy is checked against the trusted policies, directly or through other policies with issuers.

This profile is the key element upon we have based our work to define an infrasctructure for management of delegation policies, as described below.

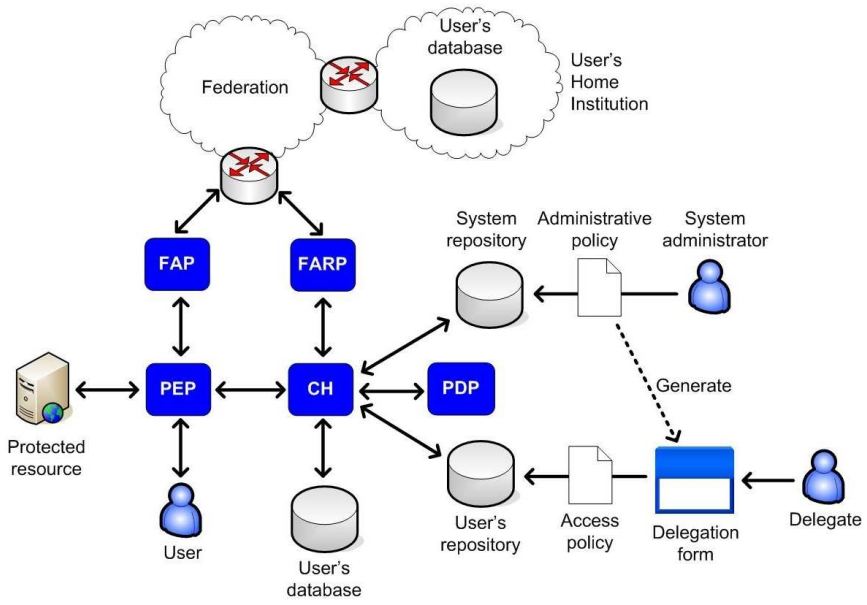


**Fig. 1** Proposed architecture

## 3 Delegation infrastructure

### *3.1 Architecture*

Usually, in the federated environments, the authentication of roaming users is carried out in their home institutions while the authorization to access a specific service is done at the remote institution based on some attributes associated with them. Thus, the federation offers common services to the member institutions to authenticate roaming users and to request their associated attributes through the federation. From this point of view, we can consider that, in a general way, the federation offers two different entities: a *Federation Authentication Point (FAP)* which authenticates a remote user and returns to the user a handle (identifier), and a *Federation Attribute Requester Point (FARP)* which returns the attributes associated with the user identified by the handle. Specifically, the work presented in this paper is based on the DAMe project [1], where authentication is based on eduroam [5] and the attribute requests are carried out through eduGAIN [4].As Figure 1 shows, the FAP in our architecture is provided by eduroam while the FARP is provided by eduGAIN. Both services forward the corresponding requests to the user's home institution, where they can be answered properly.

The figure also shows a *Policy Enforcement Point* or *PEP*, which is responsible for protecting the resources and enabling the access only to authorized users, and a

*Policy Decision Point* or *PDP*, which takes the authorization decisions based on the XACML policies and the user's attributes. Finally, we can see the *System and the User's repositories* that contains the policies generated by the system administrator and the delegates respectively. In the example described previously, the protected resource is the network, and the PEP is the access point (AP) that is providing wireless access.

### *3.2 Policy management*

The first type of policy we have to define in our system is an administrative policy expressing the delegation of the administrative rights to the delegate. It is created by the system administrator in the usual way, although to simplify the task, some XACML editor can be used [7, 2]. This policy should specify the delegate by means of the identity, or in a more general way, specifying some attributes that must be held. An important advantage of defining the delegate by means of attributes is that the policy can previously exist in the system, and the only step to take is the issuance of the appropriate attributes for the delegate. Moreover, this policy can specify some network properties to be enforced during the network connection and some access conditions for the final users, by means of obligations and the condition elements. In our example, the administrator assigns the *meeting_delegate* attribute to the meeting organizer to allow him to create the proper policies, and specifies a *high* level of QoS. However, he does not specify time conditions because he does not know the meeting schedule.

Once this policy is defined, the delegate is able to create the access policies. He is a common user of the institution without specific knowledge about the policy language, so instead of building this policy from scratch, he can use some kind of form or template. Therefore, we need a *Policy Management Tool (PMT)* to build the access policy form. This tool can be an XSLT transformation that extracts the appropriate data from the delegation policy to generate the form. In this way, the delegate only has to fill in this form to create the access policy. Additionally, this form must include an option to refine the network properties specified by the administrator, but checking the properties specified by the delegate against the others specified by the administrator. We have to take into account that the access to this form must be restricted to the the proper delegates. Therefore, the PMT must also generate a XACML policy to control the access to the policy itself. Finally, the administrative policy and the access control policy are stored in the system repository, Figure 2 on the left. In the meeting example, after the delegate is successfully authenticated in the system, he can fill in the form to generate the access policies. In the form, he also can specify the time allowed to access to the network from 9h to 18h.

There is still an open issue with the management of the access policies, because somebody has to delete them from the repository when they become invalid, for example when the meeting is finished. Despite the system administrator might be responsible for that task, the sense of administrative delegation is to free the administrator of doing this kind of tasks. Another option is to assign that responsability to the delegate. However, he is a common user not worried about these policy manage-

ment tasks, and therefore he can forget to do it. As a solution, the access policy form can force the delegate to specify the validity time for this policy. Then, as Figure 2 shows on the right, when the policies are stored in the repository, the max validity date is also specified. Thus, the list of expiration dates can be checked periodically to automatically delete the policies that have expired.

Once the policies have been successfully generated, meeting attendants can try to access to Internet. First, rhe AP authenticates them by means of the FAP. Later, their attributes are revocered via the FARP and an authorization request is sent to the PDP, but an administrative request is also needed to check whether the issuer is a right delegate. If all the authorization decisions are successful and the time condition is satisfied, the AP enables the Internet access to the user with the specified QoS. Once the meeting is finished, the access policies are deleted automatically from the user's repository because they become invalid. Finally, the system administrator deassigns the *meeting_delegate* attribute.

## References


1. *DAMe Project web site*. http://dame.inf.um.es.
2. *UMU XACML Editor Home Page*. http://xacml.dif.um.es.
3. A. Anderson et al. *EXtensible Access Control Markup Language (XACML) Version 1.0*, February 2003. OASIS Standard.
4. D.R. López et al. *Deliverable DJ5.2.2,2: GÉANT2 Authorisation and Authentication Infrastructure (AAI) Architecture - second edition*, April 2007. GN2 JRA5. GÉANT 2.
5. T. Kersting et al. *Deliverable DJ5.1.5,2: Inter-NREN Roaming Infrastructure and Service Support Cookboook - Second Edition*, August 2007. GN2 JRA5. GÉANT 2.
6. E. Rissanen and B.S. Firozabadi. *Administrative Delegation in XACML - Position Paper*, 2004.
7. M. Sánchez, G. López, O. Cánovas, and A.F. Gómez-Skarmeta. Using Microsoft Office InfoPath to Generate XACML Policies. In *Proceedings of the International Conference on Security and Cryptography (SECRYPT)*, 2006.
8. OASIS Access Control TC. *XACML v3.0 Administration and Delegation Profile Version 1.0*, October 2007. OASIS Working Draft.


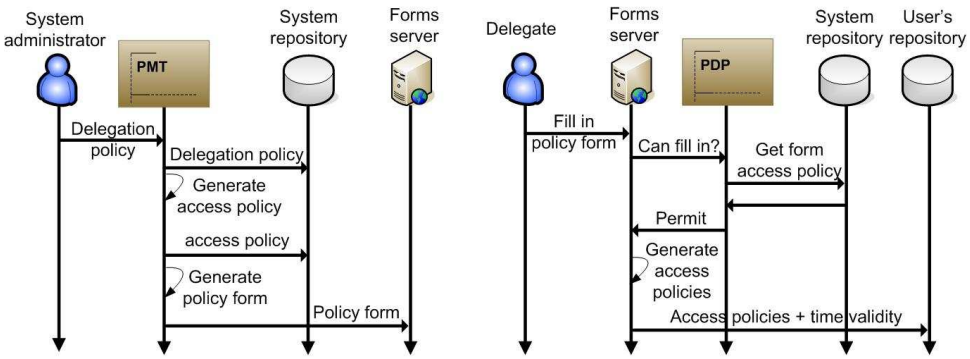


**Fig. 2** Policy management