Extending XACML for Open Web-based Scenarios

Claudio A. Ardagna[1], Sabrina De Capitani di Vimercati[1], Stefano Paraboschi[2],
Eros Pedrini[1], Pierangela Samarati[1], Mario Verdicchio[2]
[1]DTI - Università degli Studi di Milano, 26013 Crema, Italia
firstname.lastname@unimi.it
[2]DIIMM - Università degli Studi di Bergamo, 24044 Dalmine, Italia
{parabosc,verdicch}@unibg.it

Abstract

Traditional access control solutions, based on preliminary identification and authentication of the access requester, are not adequate for open Web service systems, where servers generally do not have prior knowledge of the requesters. In this paper, we provide some extensions to the eXtensible Access Control Markup Language (XACML), which is the most significant and emerging solution for controlling access in an interoperable and flexible way, to make it easily deployable and suitable for open Web-based systems.

1 Introduction

Open Web service systems, in which servers do not normally have any prior knowledge of users, call for a different solution from a traditional access control based on preliminary identification and authentication of requesters. The solutions proposed so far are in most cases logic-based and, although expressive, they hardly ever result applicable in practice, where simplicity, efficiency, and consistency with consolidated technology play a fundamental role [4, 5]. Although notably widespread in the research community and the industry, and likely the most significant proposal to date, the eXtensible Access Control Markup Language (XACML) [6] still suffers from some limitations when it comes to its capabilities in supporting the requirements of open Web-based systems. Using XACML standard extension points is however possible to define new functions, data types, and policy combination methods, thus exploiting the language's flexibility to adapt it to several different needs. In this paper, we propose a profile with a specific focus on those aspects of open world scenarios that are not currently supported by the standard. The novel concepts that access control in open Web services should include are going to be introduced together with guidelines on the modifications that need to be applied to the current XACML language and architecture to include the proposed extensions. We then illustrate how a support for certified information, abstractions, recursive reasoning, and dialog management enabling an interactive access control between clients and servers based on incremental releases of data and request for data can be deployed in XACML.

2 Deployment in XAMCL

XACML is not suitable for supporting the definition of access control policies in open scenarios, since it has some significant limitations. First, traditional XACML policies allow for the definition of generic boolean conditions referring to the different elements (e.g., subject, object, action) of a policy, thus enforcing an attribute-based access control. However, no support for expressing and reasoning about credentials is provided. Second, XACML neither provides the basis to reason about existing information, thus deriving

new concepts, nor an infrastructure to support recursive conditions like the features offered by logic-based policy languages. Third, XACML implicitly assumes that the engine that enforces access control decisions has all the necessary information to complete the evaluation process. Such an assumption is not realistic in most cases, and we counter it by calling for an infrastructure to manage the dialog between the involved parties. In the following of this section, we discuss how these limitations can be counteracted with the addition of novel features to XACML having a limited impact on the original specification, towards an extended XACML framework suitable for open Web services.

2.1 Credentials

Although designed to be integrated with the Security Assertion Markup Language (SAML) [1] for exchanging security assertions and providing protocol mechanisms, XACML lacks a real support for considering and reasoning about digital certificates, in particular, for expressing conditions on certified properties, and on properties of the certificates themselves, to which we refer to as metadata (e.g., the certificate type, or its issuer). XACML currently supports attribute-based access control; we aim at extending it to support also credential-based access control. To represent and manage credentials in XACML, and related properties and conditions on them, we need to model conditions on metadata and conditions on the attributes separately. Attributes in the credentials can be treated as any other property and need only to be associated with the proper certification. To do that with minimal impact on XACML, we reuse the *Issuer* attribute in the `SubjectAttributeDesignator` element. In particular, an occurrence of a credential attribute in the subject expression will translate into an element `SubjectAttributeDesignator`, where attribute *AttributeId* is equal to the attribute name, and attribute *Issuer* refers to the metadata. A new XML schema is then introduced to represent credential metadata. The schema has a root element `certifications` containing one or more elements `certification`. Each `certification` is composed by one or more alternative `group` elements, each containing restrictions on metadata.

2.2 Abstractions

Abstractions, also referred to as abbreviations, macros, or ontological reasoning in several logic-based proposals, allow for the derivation of new concepts (abstractions) from existing ones. They intuitively represent a shorthand by which a single concept is introduced to represent a more complex one (e.g., a set, a disjunction, or a conjunction of concepts). For instance, *id_document* (abstraction head) can be defined as an abstraction for any element in set {*identity_card*, *driver_license*, *passport*} of credentials (abstraction tail). A policy specifying that an access requester must provide an *id_document* can then be satisfied by presenting any of the three credentials above. To support abstraction specification in XACML, we prescribe the integration of XACML with XQuery [3], a language developed by W3C for querying XML data. Abstractions can be defined via XQuery functions and referenced in XACML conditions as follows. A new XML schema defines abstractions, where a root element, called `abstractions`, includes a set of single abstractions. An XQuery function, called *expansion function*, takes in input the abstraction head and produces in output the abstraction tail.

2.3 Recursive conditions

Recursion plays a fundamental part in the representation of restrictions on how authorities and, more in general, trusted parties delegate the ability to issue credentials. The delegation can be seen as a certification of the capability of another party to create credentials on behalf of the delegator. In distributed systems characterized by a complex architecture, delegation is a feature that increases flexibility and allows for a simple way to issue credentials, particularly in an open environment. In such systems, specifications of restrictions in delegation are needed, and the support for recursion in the policy language can be exploited

Table 1: From formal concepts to XACML

Basic Constructs	**XACML**
Certification metadata	⟨`certification`⟩ ... ⟨/`certification`⟩
Credential conditions	Attribute *Issuer* in element ⟨`SubjectAttributeDesignator`⟩
Abstractions	XQuery functions
Recursive statements	XQuery functions
Dialog management	Attribute *Disclosure* in any of: - element ⟨`Condition`⟩ in XACML - element ⟨`Apply`⟩ in XACML - sub-elements of ⟨`group`⟩

to specify conditions on data with a recursive structure. As for abstractions, we propose to use an XQuery engine to manage recursion in XACML. Again, recursive conditions are defined via recursive XQuery functions. These functions are then embedded and referenced in the policies, without changes to the XACML language, to define policy conditions based on recursive concepts (e.g., the supervisor concept in a business hierarchy). These functions take in input the XACML context, and produce new information to be used in policy evaluation. As a consequence, XQuery offers recursive reasoning and it allows for the creation of additional attributes to be used in the evaluation of the XACML policies during the policies evaluation process.

2.4 Dialog

The introduction of dialog between the involved parties introduces several advantages, such as enabling the server to communicate which information is needed to evaluate a policy, which in turn allows the access requester to hand over only the necessary credentials, instead of her whole set, as in the current XACML proposal. The access control process thus would become able to operate without an a-priori knowledge of the requester [2]. Extending XACML with dialog management not only would avoid the simple evaluation to indeterminate of all those cases for which the server is missing information, but it also permits to tackle the issue of the privacy trade-off between providing the whole set of credentials (on the access requester side) and disclosing the whole access control policy (on the server side). Such result can be achieved by attaching a disclosure attribute to every condition in an access control policy. Such attribute indicates what type of disclosure policy is associated with the condition, and it is then enforced by hiding from the access requester the information that cannot be released according to the specified disclosure policy. The more (less) of an access control policy is disclosed, the smaller (bigger) is the quantity of information in terms of released credential that will have to be provided by the access requester. Differently from the extensions provided to support previous concepts, dialog management requires a change in the XACML language for representing the disclosure policies associated with conditions. Each condition appearing in a XACML policy is associated with a disclosure policy represented through a new attribute *Disclosure*. This attribute is added to those elements used for representing the conditions: elements `Condition` and `Apply` in XACML, and each sub-element of element `group` in the credential schema. The admissible values for the *Disclosure* attribute are: *i) none*, nothing can be disclosed about the condition; *ii) credential*, only the information that there is a condition imposed on some metadata/attributes can be disclosed; *iii) property*, only the information that a property needs to be evaluated can be released; *iv) predicate*, both the information that a property needs to be evaluated and the related predicate can be released; *v) condition*, all information about the condition can be disclosed.

Table 1 summarizes the mapping between the novel concepts and the changes introduced to XACML.

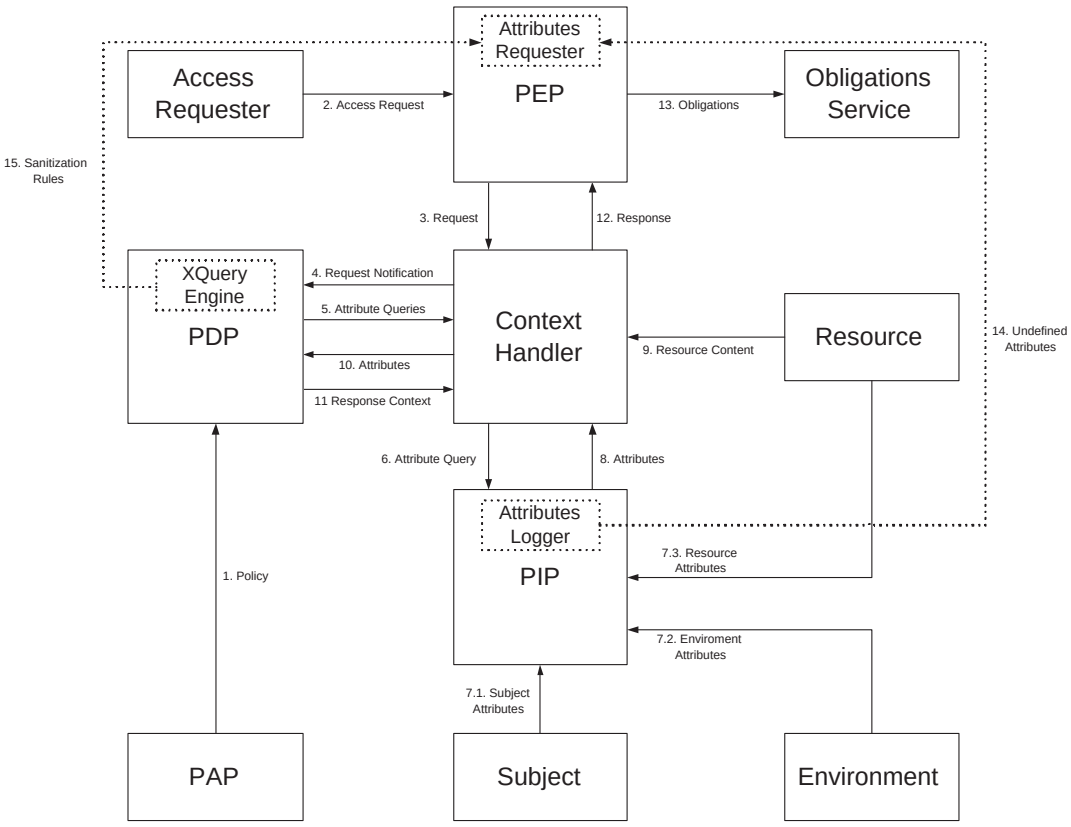

Figure 1: Extended XACML architecture

3 Extended XACML Architecture

The integration of the above novel concepts and new functionalities within the existing XACML specification requires some extensions to the standard XACML architecture. The standard architecture is comprised of functional components interacting to take an access control decision, which are depicted in Figure 1 with solid lines. Our proposed extensions are shown with dashed contours.

The *Policy Enforcement Point* (PEP) receives all access requests and enforces access decisions. The *Policy Decision Point* (PDP) computes the access decision by retrieving the policies applicable to an access request from the *Policy Administration Point* (PAP). The PDP needs all the relevant information in advance for the decision process, and the *Context Handler* manages accordingly the data directly provided by the requester. Also, the *Context Handler* works as an interface to access relevant additional information, such as the one retrieved by the *Policy Information Point* (PIP), in the form of attribute values about the subject, the resource, or the environment from the system information storage (e.g., a DBMS).

The enhancement of XACML with credentials calls for changes in the PDP component, which needs to be extended to support the evaluation of conditions regarding restrictions on the certification mechanisms. The PDP is further extended by an XQuery engine to evaluate recursive conditions. The XQuery engine is then responsible to access the Context Handler, thus retrieving all relevant attributes for policy evaluation. Since abstractions can be defined by means of XQuery functions, which take abstract concepts (e.g., *id_document*) in input and provide their expansion (e.g., the set {*identity_card*, *driver_license*, *passport*}) as output, the XQuery-based extension is used for the definition and evaluation of XACML conditions involving abstractions. When the PDP needs to reason on abstractions, it extracts the definition of the

abstractions stored in the XACML policies or profiles, and processes them in the policy under evaluation. The integration of credentials, recursive conditions, and abstractions in XACML only requires to expand the functionalities of the PDP, without modifying the communication flow of the standard architecture. The communication flow needs to be modified to take dialog management into account. In particular, a more complex PIP is needed to determine and store all the attributes that are not available at evaluation time and must be requested to the client (*Attributes Logger*). To manage such a case, resulting in an indeterminate evaluation returned by the PDP to the PEP via the Context Handler, the standard PEP must be extended with a direct communication channel to the PIP, to retrieve the list of missing attributes. Through such a channel, the PEP collects all the information requests that need to be forwarded to the user to complete the evaluation process (*Attributes Requester*). Before sending these requests, the PEP retrieves from the PDP the conditions associated with each missing attribute, and the disclosure policies relevant to such conditions. The dialog also supports XQuery-based abstractions by providing the requester with a request for data after the abstractions have been expanded. After collecting all conditions for which some attributes are missing, and applying the disclosure policy, a response that calls for additional information is returned to the requester.

4 Conclusions

We presented possible extensions to the XACML language and architecture for fully supporting the requirements of an open Web-based scenario. The extended XACML language and architecture have been designed to support several novel functionalities, including credential-based restrictions, abstractions, recursive conditions, and dialog management, with minimal impact on the standard.

References

[1] A. Anderson and H. Lockhart. *SAML 2.0 profile of XACML*. OASIS, September 2004.

[2] C.A. Ardagna, J. Camenisch, M. Kohlweiss, R. Leenes, G. Neven, B. Priem, P. Samarati, D. Sommer, and M. Verdicchio. Exploiting cryptography for privacy-enhanced access control: A result of the PRIME project. *Journal of Computer Security*, 2009. (to appear).

[3] S. Boag et al. *XQuery 1.0: An XML Query Language*. World Wide Web Consortium (W3C), 2007.

[4] P. Bonatti and P. Samarati. A unified framework for regulating access and information release on the Web. *Journal of Computer Security*, 10(3):241–272, 2002.

[5] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, and P. Samarati. Access control policies and languages in open environments. In T. Yu and S. Jajodia, editors, *Secure Data Management in Decentralized Systems*. Springer-Verlag, 2007.

[6] T. Moses. *eXtensible Access Control Markup Language (XACML) Version 2.0*. OASIS, 2005.