Policy-based Access Control in Practice

Phil Hunt, Rich Levinson, Hal Lockhart, Prateek Mishra
Oracle Corporation

1. OpenAz: A framework for Building and Deploying XACML PEPs

Increasingly, there is a consensus that access control decisions should be *externalized* from applications or services to a policy engine implementing a policy decision point (PDP). The policy decision point evaluates policies relevant to the resources that are being accessed and determines whether access should be allowed. Evaluation may require policies from multiple sources to be combined and jointly evaluated. The key benefit of this architecture is that changes to access policies can be made without changes to the service or application.

There are a variety of policy engines available in the market – these include some based on XACML, logic and a variety of proprietary rules-based engines implemented by vendors or by enterprises. While a policy engine should be thought of as a single logical entity, many vendors provide deployments in which a single policy engine can be distributed in multiple locations to meet performance and latency required by many business scenarios.

During several XACML interoperability events [XACMLInterop] organized by OASIS, as well as a joint workshop on Authorization Use-Cases [AzUseCase] organized by the Burton Group and the Liberty Alliance certain gaps and missing technologies were identified.

The first has to do with enabling support for performant and flexible creation of PEPs in a variety of contexts. From the perspective of an enterprise, the ability to embed PEPs in applications and services built using different technologies is a key part of the value offered by the policy-based authorization model. XACML does offer language independent definitions for authorization request and response which could serve as the basis for constructing PEPs. But the lack of explicit language bindings – Java, C++, PHP and so on - means that it is difficult to create PEPs with performance appropriate for varying business scenarios. Thus, as part of the OpenAz open source project [OpenAz], we have contributed a draft Java Authorization API [JavaAPI] to the XACML TC.

An important consideration here is that we are not requiring policy engines themselves to be based on XACML. We recognize a wide range of existing policy engines, and our goal instead is to connect with PEPs via a standard interface for authorization decisions based on XACML. Of course, there are some business contexts where the use of a native XACML policy engine can be of business value but this is not a requirement of our approach.

The second is motivated by the separation and independence of security policy from both the authorization service and applications in our model. Observe that XACML does not require a fixed set of attributes as part of the authorization decision process. This raises the question about the attributes required to fully evaluate the security policy at some point in time. The requirement is for a representation that can express the set of attributes required by a policy or the set of attributes added by a PIP associated with the Authorization service or an application. This information can be used at deployment to determine whether needed attributes are available and, if some are missing, can provide the basis for updating components so that they become available.

Authorization API

XACML does define an abstract syntax in XML, for conveying the inputs and outputs of a policy decision. XACML also defines a network protocol, the SAML profile of XACML, for making requests of a policy engine located in a remote server based on the abstract XML syntax.

However the XACML standard does not define any standard API which can be used to request a policy decision. The XML format it specifies for inputs and outputs is not very convenient or efficient to invoke a policy engine which is located within the same address space. In most cases the input data required is already available in some format convenient to use in the given programming language and the policy engine uses a similar format internally. Serializing the data into XML and back out simply adds extra programming effort and reduces performance. For this reason the XACML standard does not mandate the use of the XML format, only equivalent functionality.

 A standard API would also be useful when a remote policy engine is being invoked via a network call. It makes little sense for every application contain the code to produce a properly formatted network request. A more reasonable design is to provide a library that can be called to handle the messaging details. This also would be facilitated by a standard Authorization API.

Two features of the API are worth highlighting: support for bulk authorization, wherein multiple authorization decisions can be packaged into a single request, and, support for a limited form of "scoped" authorization query.

Expressing bulk authorization is important as it allows a PEP to communicate interest in obtaining decisions for an entire set of authorization questions in a single invocation. This information allows the authorization service and policy engines to provide enhanced performance in responding to the decision request.

The "scoped" authorization query is motivated by a class of authorization use-cases, wherein it is necessary to discover the rights and entitlements of a subject within a certain context. A good example is a portal application, which is required to display only the

services appropriate for the user who has authenticated to the portal. The "scope" information – which takes the form of an uninterpreted identifier - serves as short-hand for a set of interactions potentially available to all users; the response reduces this set to those explicitly accessible to the user who has authenticated to the system and further explicitly returns the set of interactions as part of the result.

Providing Necessary Attributes

Access control policy languages, such as XACML feature a policy language syntax and processing rules for evaluation as well as a format for providing necessary input data and referencing it from the policy language. The language syntax and processing rules are encapsulated from all other components of the access control system and protected applications. The actual policies evaluated by a given deployed system are also encapsulated from all other components except for the tools used to construct them. And in fact, these systems may not even have the ability to evaluate policies, merely to produce them from human input or alternative policy representations.

This encapsulation has a number of benefits, discussed elsewhere, but it leads to a conundrum. In order for policies to be evaluated and an access control decision made, the input data must be provided. However this data must be provided at the time of the decision by software components which are unaware of the policies currently in force or the rules for evaluating them. Of course the PDP can make attribute requests to obtain the values referenced by policies which are missing from the request, however this will increase decision latency and usually reduce overall throughput of the PDP.

If there was no cost providing information or if the total range of inputs to policy decisions was small, callers could simply provide all available information with each request. Many authorization decision systems operate in this way today. For this reason we are proposing a data format which should allow callers of a PDP to provide information more closely tailored to policy needs, while not breaking the encapsulation of the PDP.

This format is referred to as the Attribute Manifest File [AMF] and a draft has been published to XACML TC. This XML file consists of a series of entries which describe attributes which are available and may be used in Authorization decisions. Entries may consist of three kinds of information:
- Identifying Information: attribute name, issuer, datatype;
- Value Information: legal range of values, enumerated values; and
- Retrieval Information: location, access method, database keys, etc.

2. *Identity Governance Framework: privacy-aware access control for identity data*

In modern economies, information about citizens and consumers is held by multiple authorities and goverment departments. The conditions under which such data can be released is based on citizen consent and by legal agreements under which data was

obtained. However, there is a lack of standards and architecture to support the managed release of identity data to services and applications that require such data.

Identity Governance Framework [IGF] is a framework that supports auditable and privacy-aware interaction between *identity authorities* and *business processes* that require identity data.

In the IGF vision, there is a world-wide network of authorities that provide identity data about subjects. At the other end, there are business services or relying parties that require identity data in order to be effective. IGF makes no technology assumptions concerning how the identity data is aggregated and obtained or transferred from authorities to a business service. For example, data may be delivered from the user via a form, or from an IdP within a SAML assertion, or from databases and directories.
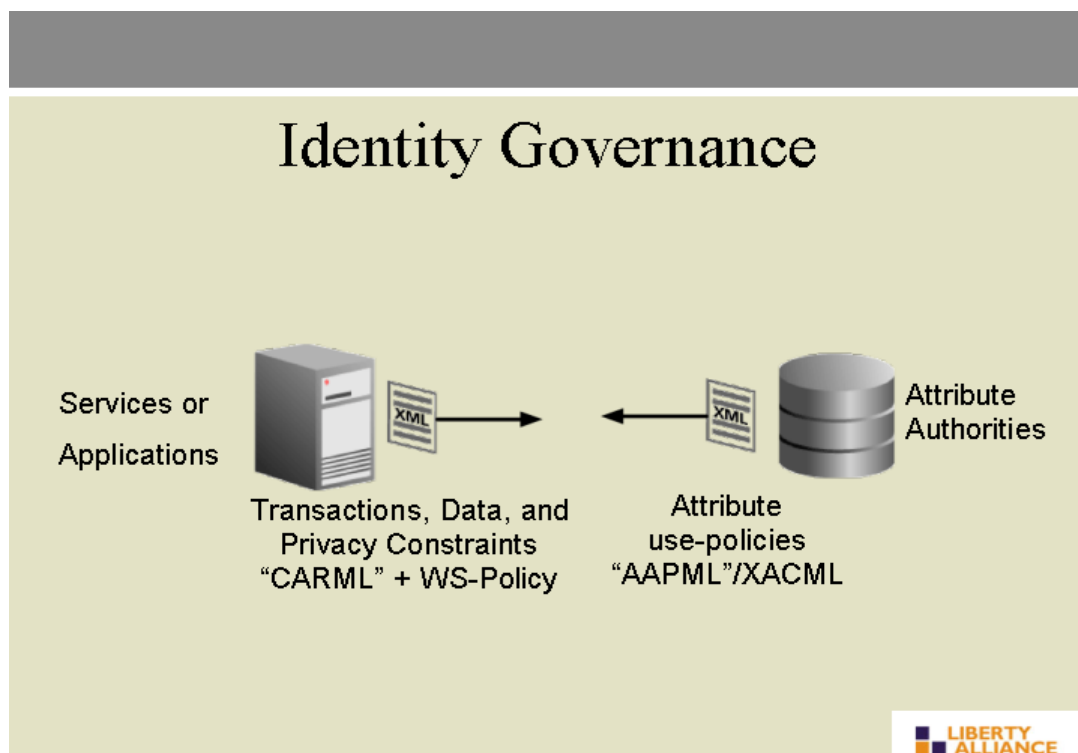
Attribute Authority Markup Language or AAPML (pronounced "app"-mul) is a profile of XACML suited to describing conditions under which identity data should be released to a service or application. A draft document [AAPML] describing the profile has been published to the XACML TC.

AAPML rules includes considerations of:

(1) The relationship between the end-user seeking the data and the user about whose identity information is being sought. AAPML rules allow the specification of required relationship between these identities, for example, that they be identical or related in some particular way (e.g., friend or manager).

(2) Whether privacy constraints governing the release of identity data are being met.

IGF includes a language of privacy constraints described in [PrivacyConstr]. Privacy constraints describe fundamental constraints on the propagation, usage, retention, storage and display of identity data. Using policy frameworks such as WS-Policy, authorities (custodians of identity data, end-users) and consumers (applications, enterprises) use Privacy constraints to describe composite constraints on identity data.

When a request for identity data is received by an attribute authority, the access control system examines the privacy constraints bound by the requestor to the desired attribute. This is interpreted as a 'commitment' the requestor is making with respect to its actions should it receive the attribute. Conversely, when an authority binds privacy constraints to the access control rules for an attribute, a constraint is interpreted as an 'requirement' attendant upon the recipient.

Access to an attribute is granted only if the privacy "commitments" made by a requestor are consistent with the constraints bound to the attribute data by the authority.

(3) Whether consent has been obtained from the user or delegate concerning the release of identity data. The requirement that specific consent agreements be met is modeled as a type of XACML obligation. A *consent processor* analyzes consent obligations returned by the access control system and determines whether relevant consent information is available and current. If consent cannot be verified, additional information about the required consent agreement can be returned to the requestor.

AAPML rules can be published by authorities to PDPs and enforced by PEPs located at enforcement points located within components providing access to personal data. Use of a single XACML-based policy framework for access control and privacy supports an integrated view of the rules governing the release of identity data.

REFERENCES

[XACMLInterop] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#INTEROPS
[AzUseCase] Catalyst Concordia Workshop on Authorization and Entitlement, June 2008
http://projectconcordia.org/index.php/Catalyst_Concordia_Policy_Workshop_2008
[OpenAz] http://www.openliberty.org/wiki/index.php/Main_Page#OpenAz
[JavaAPI] http://lists.oasis-open.org/archives/xacml/200907/msg00019.html
[AMF] http://lists.oasis-open.org/archives/xacml/200907/msg00019.html
[IGF] http://www.projectliberty.org/strategic_initiatives/identity_governance
[AAPML] http://www.oracle.com/technology/tech/standards/idm/igf/pdf/IGF-AAPML-spec-08.pdf
[PrivacyConstr]
http://www.projectliberty.org/liberty/resource_center/specifications/igf_1_0_specs