GTAuthzFramework

Globus Toolkit Authorization Framework

The *Globus Toolkit Authorization Framework* is designed to handle different attribute mechanisms in a consistent manner and is able to combine authorization decisions from many different sources to yield a single access decision for each invocation request.

Attribute Collection

When a client invokes a request with a service, that service may have to consider many different identity and attribute formats, like X.509 EECs, X.509 Attribute Certificates, SAML Attribute Assertions, LDAP attributes, Handle System attributes, and Configuration properties.

As it is very common that requests by a client are made on behalf of other parties, some of those attribute values do not necessarily apply to the requester, but can apply to the intermediates or the originating user in the delegation chain.

Furthermore, the attributes can arrive at the service in a number of different ways. Some attribute assertions are "pushed" by the requester, where the assertion is either embedded in the requester?s proxy certificate or in the SOAP header, as in VOMS or CAS.

Other attributes are "pulled" by the service from attribute services, like LDAP, SAML Attribute Query compatible services like the Shibboleth Attribute Authority, or the Handle System. Note that each of the pull-mechanisms uses different protocols.

Lastly, attributes are also locally stored in (configuration) files on the service?s side.

The validation of the attribute binding information is also dependent on the assertion format and how the information was received. Some attribute bindings are asserted through public key signatures, while others are received unsigned but embedded in protected messages or received over authenticated channels.

Finally, the attributes names and values have to be considered within the context of their definition and within the context of the issuer. Besides the vocabulary, semantics, and ontology that apply to the attribute bindings, it is also important to understand clearly whether the assertion is only valid in the local context of the issuer or in a global context that requires additional authorization during the validation.

In order to manage the attribute collection in a consistent manner, the Globus team is in the process of developing a general attribute framework. Its purpose is to accept and validate the various attribute assertion formats and mechanisms, to group all the attributes that apply to the same entity together, to translate the names and values into a single format, and finally to make the attribute collections available to the subsequent authorization decision processing phase.

Authorization Mechanisms

As was the case for attribute collection, the processing of the authorization policy enforcement is a similar challenge because of the fact that many formats and mechanisms have to be supported.

The applicable authorization policy can come from many different sources, like the resource owner, the resource domain, the requester, the requester?s domain, the virtual organization, or intermediaries.

Authorization decisions can be evaluated within the same hosting environment as the policy enforcement point, or can be evaluated by external authorization services.

We have the common delegation-of-rights scenario where one subject can empower others to work on her behalf through the issuing of policy statements. As a consequence, there can be multiple policies and decisions that have to be combined to yield a single decision about the access rights of the requester.

The requester can push some of these policy statements or decisions as authorization assertions, which have to be evaluated by the resource owner. Proxy certificates are essentially examples of such authorization assertions. CAS uses SAML authorization decision assertions that are either embedded in proxy certificates or communicated in the SOAP header.

There are many different mechanisms and languages used to express authorization policies, like grid-mapfiles, proxy certificates, SAML authorization decision assertions, CAS policy rules, XACML policy statements, and simple ACLs.

Authorization Decision Evaluation

After all the attributes and authorization assertions are collected, and internal and external authorization services are identified, the authorization decision for the access request can be determined.

In order to be able to deal with different authorization mechanisms, the authorization framework uses a PDP abstraction. Each mechanism essentially needs its own custom decision evaluator that understands the intrinsic semantics of the policy expressions. The PDP abstraction uses a common interface to interact with the different mechanism-specific authorization decision evaluators. This common interface is mimicked after the XACML request context interface, which essentially presents the decision request as a collection of attribute values for the subject, resource and action. The evaluated decision result can have the values of permit, deny or not-applicable. Note that the PDP?s decision is associated with either the issuer of the policies that were evaluated or with the identity associated with an (external) authorization service.

For each received authorization assertion and for each authorization service, a mechanism-specific PDP instance is created. As each of those PDP instances is queried through the same interface to evaluate authorization decisions, the mechanism-specific details are all hidden behind the abstraction.

A separate Master PDP abstraction is used to combine all the different decisions from all the different PDP instances in such a way that a single decision reflects the overall evaluated policy. In essence, this Master PDP queries the different PDP instances about the access rights of the requester and potential delegates, and searches for valid delegation decision chains that originate from the resource?s owners policy and end with a statement that speaks on the access rights of the requester.

Note that through the use of PDP abstractions, the framework is able to evaluate decisions about delegated access rights for the requester, without the need for explicit support of delegation in the policy languages used in the authorization mechanisms.