OpenAz: XACML PEPs for Attribute-based Access Control

Rich Levinson

Hal Lockhart

Prateek Mishra

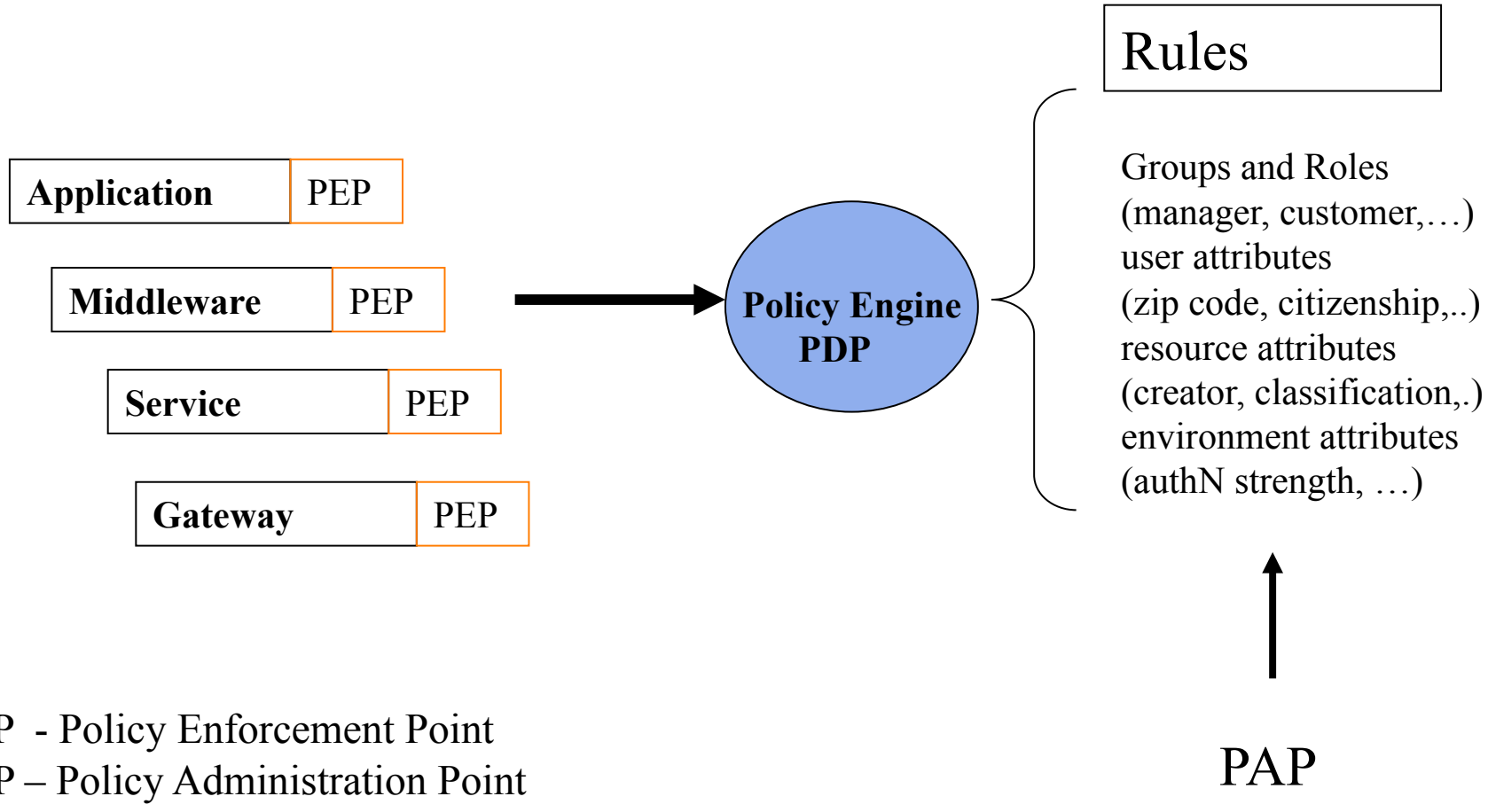
Oracle Corporation, July 2010

kantara INITIATIVE™

Glossary

- ABAC: A policy-based access control solution that uses attributes assigned to subjects, resources or the environment to enable access to resources and controlled information sharing [NIST2010]
- Policy Engine: rules-based engine that implements a policy decision point
  - Many commercial policy engines available
  - Specialized policy engines include domain knowledge
  - Some policy engines accept or generate rules using the XACML policy language

Externalized Attribute-based Access Control



PEP - Policy Enforcement Point
PAP – Policy Administration Point
PDP – Policy Decision Point

What is a XACML PEP?

- Interacts with PDP using XACML authorization request and response model
  - XACML provides XML definition of request/response
    - XACML SAML profile is a concrete protocol instance
  - Opportunity for other language and protocol bindings
- Authorization Request is a set of attributes – name and value pairs
  - Subject, Environment, Resource, Action
  - Attributes can be added by multiple entities
    - Application, container, proxy, middleware, policy engine, ….

Authorization Response

- Supports feedback from policy engine to PEP
- Obligation – additional conditions that must be enforced by the PEP before access is allowed
  - Logging, privacy, user-interaction
- Missing attributes
  - PEP can discover required attributes dynamically
  - Policy changes may result in new attributes being required

OpenAz Goals

- Provide consistent model for applications and middleware to invoke access control
  - XACML PEP can be embedded in a variety of contexts
  - Support for finding attributes on an as-needed basis
  - Encourage creation of other language/framework bindings
- Reference implementation for Java AzApi interface
  - Java binding for XACML request-response protocol
- Explain how XACML Java AzApi interface can be mated with third-party policy engines
  - Existing policy engines can implement this interface
  - Support efficient processing as providers can implement caching and other proprietary magic
  - Details of local vs. remote processing hidden by the interface

Available OpenAz Components

- Definition of Java AzApi Interface
  - Includes Java Construct layer
  - Submitted to XACML TC for standardization
    - Joint work with Cisco and others (RSA)
- Implementation of AzApi with SUN XACML library
  - Available for use today
- XACML Policy-creation Tool
  - Simplifies creation of XACML policy

Java AzApi: XACML Abstraction Layer

- A set of interfaces that enables a Java module to supply and consume all the required info for submitting a XACML request and for receiving a XACML response, respectively.
- The main API is:

```
AzResponseContext azRsp =
    AzService.decide(AzRequestContext azReq);
```

Java AzApi: objects

- **AzService**: main impl from an AzApi provider; OpenAz provides ref:
  (`org.openliberty.openaz.pdp.provider.`**`SimpleConcreteSunXacmlService`**)
- **AzRequestContext**, **AzResponseContext**: provided by OpenAz, optionally may be implemented by provider
  AzEntity<AzCategoryId, Enum<T>>: a collection of AzAttributes in a specific AzCategoryId (Subject, Resource, Action, …)
- **AzAttribute<AzCategoryId, Enum<T>>**: a collection of XACML Attribute metadata associated with a specific AttributeId
- **AzAttributeValue<AzDataTypeId,Enum<U>,V>**: a XACML DataType and corresponding Java value, V, that is used to populate the DataType

Additional AzApi Features
(beyond single XACML req/rsp)

- **Multiple request/response**: a Set of AzResourceActionAssociations may be submitted with an AzEntity<Subject> and AzEntity<Environment> and a corresponding Set of AzResults is returned within the AzRequestContext and AzResponseContext, respectively.
- **AzService.query(String scope, AzRequestContext azReq)**: a Set of AzResults is returned based on a resource-specific formatted scope (ex. scope = “EngServer”, will return list of eng servers user has access to)

Additional AzApi Features (cont.)

- **Collection<AzAttribute<T>>**
  AzAttributeFinder.findAttribute(

  AzRequestContext azReqCtx,
  AzEntity<T> azEntity,
  AzAttribute<T> azAttr):

  - Applications or middleware may register one or more AzAttributeFinder objects with AzService that may be called during a “decide()” to obtain additional attributes needed to make a decision.
    The provider supplies the request context, the entity for which an attribute is requested, and an AzAttribute containing the attribute metadata.

Java AzApi: Java Construct layer

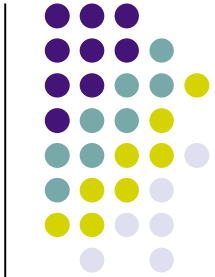
- Responds to concern that AzApi requires some knowledge of XACML specifics
  - Data types, Attribute categories and names
- Java packages or frameworks may request authorization decisions using native objects
  - E.g., Decide (user object, resource object, action object)
  - Mapping of these native representations into lower-level AzApi forms is modeled separately

Java AzApi: Java construct layer

- PepRequest pepReq =
  pepRequestFactory.newPepRequest(
  obj, obj, obj, obj):
  create a XACML request using any type of Java Objects containing subject, resource, action, and environment attributes respectively
  (ex Subject(JAAS),"read","C:/file.html",Date)
- PepResponse pepRsp = pepReq.decide();
  return a full PepResponse based on pepReq, containing boolean AzDecision and Map<String,String> of Obligations
- Mappers: custom modules used to map specific Java objects to their AzApi AzAttribute counterparts.

Download information

- Complete project (AzApi interface, reference implementation, Policy Tool,Javadoc)
  - http://openaz.svn.sourceforge.net/viewvc/openaz/ (download the GNU tarball)
- Javadoc only
  - http://openaz.svn.sourceforge.net/viewvc/openaz/azapi/doc/
- Apache 2.0 license
- Join the project !
  - http://www.openliberty.org/wiki/index.php/Main_Page#OpenAz
  - Mailing list and bi-weekly conference call



kantara INITIATIVE™