Access Control in Distributed Systems

Murat Kantarcioglu

Topics

- Overview
- SAML
- XACML

Overview

- Security for distributed systems has been widely investigated; we can distinguish:
  - Network security
  - Middleware security
  - World wide web security

Middleware security

- Past work:
  - Kerberos
  - CORBA (Common Object Request Broker Architecture)
- Current work:
  - Federated Digital Identity Management
  - Access Control and Authorization
    - XACML
    - SAML
  - Core Security Standards
    - XML Digital Signature
    - XML Encryption
- Advanced Security
  - Web services (WS) security

Middleware
Relevant Standards Bodies

- W3C
  - XML, SOAP, WSDL, XML Encryption, XML Digital Signature, XKMS
- OASIS
  - UDDI, SAML, XACML, WS-Security, WS-Policy, WS-Trust, WS-Authorization, WS-SecureConversation, WS-Federation, WS-*
    - WS-* standards developed by MS/IBM and submitted to OASIS for standardization

World wide web security

- The WWW has changed the nature of distributed computing:
  - The separation of program and data is once more abolished. Content providers embed *executable content* (*applets*) in documents to create interactive web pages that can process user input
  - Computation is moved to the client. It is thus now the client who needs protection from rogue content providers
  - Mobile code moves from machine to machine, collecting information from different places or looking from spare computer resources. Clients need protection from mobile code; mobile code may need protection from the clients it is running on
  - Users are forced to become system administrators and policy makers

World wide web security

- Relevant work:
  - Security for Java
  - Security for mobile agents
  - Intellectual property protection
    - Watermarking and fingerprinting techniques

Background Notions - XML

- eXtensible Markup Language
  - W3C Recommendation (third edition)
  - http://www.w3.org/TR/REC-xml/
- A restricted form of SGML (an ISO standard)
- Allows delivery of custom data
- Focuses on *what data is,* not what data looks like (e.g., HTML)
  - Use a Document Type Definition (DTD) or XML Schema (http://www.w3.org/TR/xmlschema-0/) to describe new syntax

Simple XML Example

```
<?xml version="1.1"?>
<note>
  <date>2004-11-10</date>
  <to>Adam</to>
  <from>Kody</from>
  <heading>Hungry</heading>
  <body>Feed me, dad!</body>
</note>
```

Background Notions - XML with DTD

```
<?xml version="1.1"?>
<!DOCTYPE note[
  <!ELEMENT note (date, to, from, heading, body)>
  <!ELEMENT date (#PCDATA)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
]>
<note>
  <date>2004-10-11</date>
  <to>Adam</to>
  <from>Jasmine</from>
  <heading>Bone</heading>
  <body>Kody stole my bone!</body>
</note>
```

Schema Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.w3schools.com"
  xmlns="http://www.w3schools.com"
  elementFormDefault="qualified">

  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
       <xs:element name=“date“ type=“xs:date”/>
       <xs:element name="to" type="xs:string"/>
       <xs:element name="from" type="xs:string"/>
       <xs:element name="heading" type="xs:string"/>
       <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

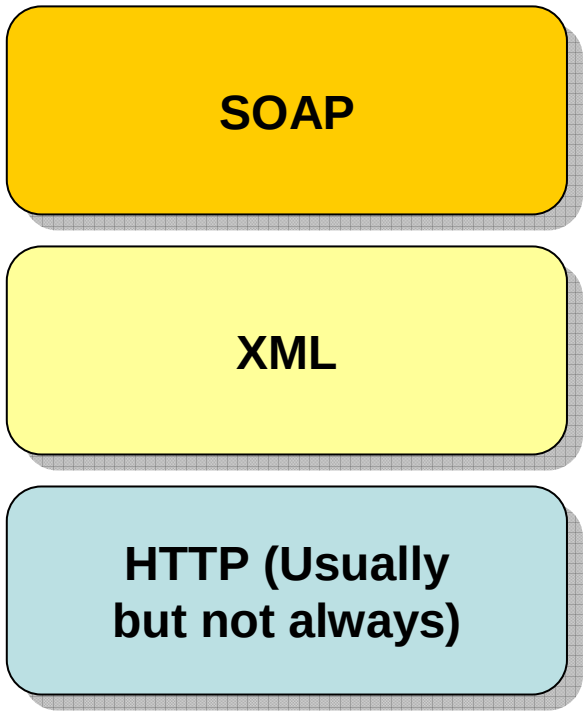
Background Notions - DOM

- Document Object Model
  - http://www.w3.org/DOM/
- Internal representation of an XML document as a tree
- Allows one to specify an element and all the data inside it as a subtree
- Also allows one to specify a search pattern over the document (e.g. XPath)

Background Notions - SOAP

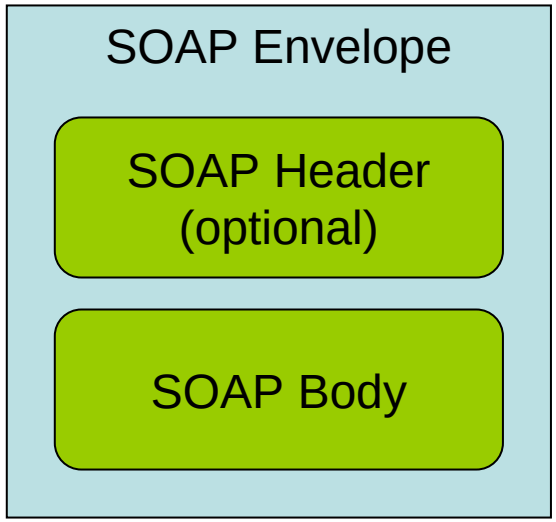
- Simple Object Access Protocol
  - http://www.w3.org/TR/soap/
- SOAP provides the definition of the XML-based information which can be used for exchanging structured and typed information between peers in a decentralized, distributed environment
- SOAP is a stateless, one-way message paradigm
- Extensible messaging framework
  - Issues such as security not part of specification, addressed as extensions

Background Notions - The Stack



Background Notions - SOAP Messages

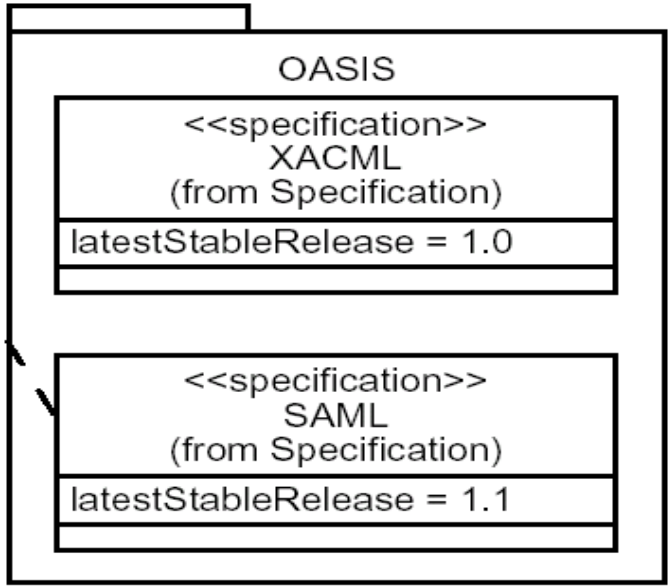
- Two main parts to the message
  - Header: Contains message meta-information
  - Body: Contains the main message



Background Notions - SOAP Example

```
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
   <n:priority>1</n:priority>
   <n:expires>2001-06-22T14:00:00-05:00</n:expires>
  </n:alertcontrol>
 </env:Header>
 <env:Body>
  <m:alert xmlns:m="http://example.org/alert">
   <m:msg>Pay the electric bill today!</m:msg>
  </m:alert>
 </env:Body>
</env:Envelope>
```

Access Control and Authorization



SAML

- Security Assertion Markup Language
  http://xml.coverpages.org/SAML-TechOverview20v03-11511.pdf
- The goal of SAML is:
  - *… to define, enhance, and maintain a standard XML-based framework for creating and exchanging authentication and authorization information.*
- Allows an organization to make assertions about security properties of a subject
  - Authentication
  - Attributes
  - Authorization decisions

SAML

- SAML is different from other security systems due to its approach of expressing assertions about a subject that other applications within a network can trust.

- The following two concepts are important in SAML
  - **Identity Provider (IdP):** The system, or administrative domain, that asserts information about a subject. For instance, the Identity Provider asserts that this user has been authenticated and has given associated attributes.

    For example: This user is John Doe, he has an email address of ***john.doe@acompany.com***, and he was authenticated into this system using a password mechanism. In SAML, Identity Providers are also known as ***SAML authorities*** and ***Asserting Parties***.
  - **Service Provider (SP):** The system, or administrative domain, that relies on information supplied to it by the Identity Provider. It is up to the Service Provider as to whether it trusts the assertions provided to it. SAML defines a number of mechanisms that enable the Service Provider to trust the assertions provided to it. It should be noted that although a Service Provider can trust the provided assertions provided, local access policy defines whether the subject may access local resources. Therefore, although the Service Provider trusts that a given user is ***John Doe –*** it doesn't mean such user is given carte blanche access to all resources. Service Providers are also known as ***Relying Parties*** – due to the fact that they “rely” on information provided by an Identity Provider (Asserting Party).

Why is SAML required?

- Four main drivers:
  - Limitations of browser cookies: most existing single-sign-on (SSO) product use browser cookies to maintain the state so that re-authentication is not needed. However, cookies cannot are not transferred among different DNS; so a different technology is required.
  - SSO interoperability: different SSO solutions are proprietary and not interoperable.
  - Web services: WS security is still being defined. It is likely that access control for WS will use authentication and authorization assertions
  - Federation: the need to simplify identity management across organizational boundaries, allowing users to consolidate many local identities into a single (or at least a reduced set) federated identity

Example Scenario of SSO

- A user has a logon session (that is a *security context)* on a website (AirlineInc.com) and is accessing resources on that site.
- At some either explicitly or transparently he is directed over to another web site (in a different DNS domain) – CarRentalInc.com
- The Identity Provider site (AirlineInc.com) asserts to the Service Provider site (CarRentalInc.com) that the user is known to it and provides the user's name and session attributes (e.g. “Gold member”).
- As CarRentalInc.com trusts AirlineInc.com it knows that the user is valid and creates a session for the user based on the user's name and/or the user attributes.
- This use case illustrates the fact that the user is not required to re-authenticate when directed over to the CarRentalInc.com site

SAML assertions

- The assertion is the main element of SAML. It is a package of information that supplies one or more statements made by a SAML authority.

- Three different kinds of assertion statement that can be created by a SAML authority:
  - **Authentication:** The specified subject was authenticated by a particular means at a particular time
  - **Attribute:** The specified subject is associated with the supplied attributes.
  - **Authorization Decision:** A request to allow the specified subject to access the specified resource has been granted or denied.

- The outer structure of an assertion is generic, providing information that is common to all of the statements within it. Within an assertion, a series of inner elements describe the authentication, authorization decision, attribute, or user-defined statements containing the specifics.

Sample SAML Assertion

```
<saml:Assertion
  MajorVersion=“2" MinorVersion="0"
  AssertionID="128.9.167.32.12345678"
  Issuer="Company.com"
  IssueInstant="2002-03-21T10:02:00Z">
<saml:Conditions
  NotBefore="2002-03-21T10:02:00Z"
  NotAfter="2002-03-21T10:07:00Z" />
  <saml:AuthnStatement
    AuthenticationMethod="password"
    AuthenticationInstant="2002-03-21T10:02:00Z">
    <saml:Subject>
      <saml:NameIdentifier
        SecurityDomain="Comany.com"
        Name="joeuser" />
    </saml:Subject>
  </saml:AuthenticationStatement>
</saml:Assertion>
```

Major Components of SAML Assertions

- Element <Assertion> specifies the basic information that is common to all assertions, including the following elements and attributes:
  - MajorVersion [Required] The major version of SAML used to express this assertion. The identifier for the version of SAML defined in the last specification is 2.
  - MinorVersion [Required] The minor version of SAML used to express this assertion. The identifier for the version of SAML defined in the last specification is 0.
  - ID [Required] The identifier for this assertion. It must be of type **xsd:ID**, and MUST be unique
  - IssueInstant [Required] The time instant of issue of the assertion
  - <Issuer> [Required] The SAML authority that is making the claim(s) in the assertion. The issuer identity SHOULD be unambiguous to the intended relying parties.

Major Components of SAML Assertions

- <ds:Signature> [Optional]: an XML signature that authenticate the assertion
- <Subject> [Optional]: The subject of the statement(s) in the assertion. There is a SAML fragment dealing with the specification of subjects
- <Conditions> element: conditions that must be taken into account in assessing the validity of and/or using the assertion
- One or more <Statement> elements:
  - <AuthnStatement>
  - <AuthzDecisionStatement>
  - <AttributeStatement>
  - Note: An assertion with no statements MUST contain a <Subject> element. Such an assertion identifies a principal in a manner which can be referenced or confirmed using SAML methods, but asserts no further information associated with that principal.

Schema Fragment for SAML Assertions

```
<element name="Assertion" type="saml:AssertionType"/>
<complexType name="AssertionType">
    <sequence>
        <element ref="saml:Issuer"/>
        <element ref="ds:Signature" minOccurs="0"/>
        <element ref="saml:Subject" minOccurs="0"/>
        <element ref="saml:Conditions" minOccurs="0"/>
        <element ref="saml:Advice" minOccurs="0"/>
        <choice minOccurs="0" maxOccurs="unbounded">
            <element ref="saml:Statement"/>
            <element ref="saml:AuthnStatement"/>
            <element ref="saml:AuthzDecisionStatement"/>
            <element ref="saml:AttributeStatement"/>
        </choice>
    </sequence>
    <attribute name="MajorVersion" type="integer" use="required"/>
    <attribute name="MinorVersion" type="integer" use="required"/>
    <attribute name="ID" type="ID" use="required"/>
    <attribute name="IssueInstant" type="dateTime" use="required"/>
</complexType>
```

SAML authentication assertion

- The <AuthnStatement> element describes a statement by the SAML authority asserting that the statement's subject was authenticated by a particular means at a particular time.
- It include following elements and attributes:
  - AuthenticationMethod [Required]: A URI reference that specifies the type of authentication that took place.
  - AuthenticationInstant [Required]: Specifies the time at which the authentication took place.
  - <SubjectLocality> [Optional]: Specifies the DNS domain name and IP address for the system entity from which the subject was apparently authenticated.
  - <AuthorityBinding> [Any Number]: Indicates that additional information about the subject of the statement may be available.

Authentication Method Identifiers

- An authentication statement with an AuthenticationMethod attribute describes an authentication act that occurred in the past.
- The AuthenticationMethod attribute indicates how that authentication was done. Note that the authentication statement does not provide the means to perform that authentication, such as a password, key, or certificate.

Authentication Method Identifiers

- **7.1.1 Password:** The authentication was performed by means of a password.
- **7.1.2 Kerberos:** The authentication was performed by means of the Kerberos protocol **[RFC 1510]**, an instantiation of the 1856 Needham-Schroeder symmetric key authentication mechanism **[Needham78]**.
- **7.1.3 Secure Remote Password (SRP):** The authentication was performed by means of Secure Remote Password protocol as specified in **[RFC 2945]**.
- **7.1.4 Hardware Token:**The authentication was performed using some (unspecified) hardware token.
- **7.1.5 SSL/TLS Certificate Based Client Authentication:**The authentication was performed using either the SSL or TLS protocol with certificate-based client authentication. TLS is described in **[RFC 2246]**.
- **7.1.6 X.509 Public Key:** The authentication was performed by some (unspecified) mechanism on a key authenticated by means of an X.509 PKI **[X.500][PKIX]**. It may have been one of the mechanisms for which a more specific identifier has been defined below.
- **7.1.7 PGP Public Key:** The authentication was performed by some (unspecified) mechanism on a key authenticated by means of a PGP web of trust **[PGP]**. It may have been one of the mechanisms for which a more specific identifier has been defined below.
- **7.1.8 SPKI Public Key:** The authentication was performed by some (unspecified) mechanism on a key authenticated by means of a SPKI PKI **[SPKI]**. It may have been one of the mechanisms for which a more specific identifier has been defined below.
- **7.1.9 XKMS Public Key:** The authentication was performed by some (unspecified) mechanism on a key authenticated by means of a XKMS trust service **[XKMS]**. It may have been one of the mechanisms for which a more specific identifier has been defined below.
- **7.1.10 XML Digital Signature:** The authentication was performed by means of an XML digital signature **[RFC 3075]**.
- **7.1.11 Unspecified:** The authentication was performed by an unspecified means.

SAML attribute assertion

- The <AttributeStatement> element describes a statement by the SAML authority asserting that the statement's subject is associated with the specified attributes.

- It is of type **AttributeStatementType**, which extends **SubjectStatementAbstractType** with the addition of the following elements
  - <Attribute>: The <Attribute> element specifies an attribute of the subject**.**
  - <EncryptedAttribute>: this element contains encrypted values; values are encrypted according to the XML Encryption Standard

SAML authorization decision assertion

- The <AuthzDecisionStatement> element describes a statement by the SAML authority asserting that a request for access by the statement's subject to the specified resource has resulted in the specified authorization decision on the basis of some optionally specified evidence.

- The resource is identified by means of a URI reference. In order for the assertion to be interpreted correctly and securely, the SAML authority and SAML relying party MUST interpret each URI reference in a consistent manner. Failure to achieve a consistent URI reference interpretation can result in different authorization decisions depending on the encoding of the resource URI reference.

- An assertion containing an <AuthzDecisionStatement> must contain the subject element (in order to bind the authorization to a subject)

SAML authorization decision assertion

- Main sub-elements of <AuthzDecisionStatement> are:
  - Resource [Required]: A URI reference identifying the resource to which access authorization is sought.
  - Decision [Required]: The decision rendered by the SAML authority with respect to the specified resource. The value is one of: Permit, Deny, Indeterminate
  - <Action> [One or more]: The set of actions authorized to be performed on the specified resource. Possible values:
    - Read: The subject may read the resource.
    - Write: The subject may modify the resource.
    - Execute: The subject may execute the resource.
    - Delete: The subject may delete the resource.
    - Control: The subject may specify the access control policy for the resource.
    - Actions can be also negated.
  - <Evidence> [Optional]: A set of assertions that the SAML authority relied on in making the decision.

SAML protocols

- SAML assertions MAY be generated and exchanged using a variety of protocols.
- The bindings and profiles specification for SAML **[SAMLBind]** describes specific means of transporting assertions using existing widely deployed protocols.
- SAML-aware requesters MAY in addition use the SAML request-response protocol defined by the <Request> and <Response> elements. The requester sends a <Request> element to a SAML responder, and the responder generates a <Response> element.
- An interesting set of protocols is represented by “queries”. Queries allows a party to require assertions concerning a given entity

SAML queries

Element <AuthnQuery>

- The <AuthenticationQuery> element is used to make the query “What assertions containing authentication statements are available for this subject?” A successful response will be in the form of assertions containing authentication statements.
- In response to an authentication query, a SAML authority returns assertions with authentication statements

SAML queries

- **Element <AttributeQuery>**
  - The <AttributeQuery> element is used to make the query “Return the requested attributes for this subject.” A successful response will be in the form of assertions containing attribute statements.

SAML queries

- **Element <AuthzDecisionQuery>**
  - The <AuthorizationDecisionQuery> element is used to make the query “Should these actions on this resource be allowed for this subject, given this evidence?” A successful response will be in the form of assertions containing authorization decision statements.

SAML threat model

- Assumptions:
  - the two or more endpoints of a SAML transaction are uncompromised, but that the attacker has complete control over the communications channel.
  - Additionally, due to the nature of SAML as a multi-party authentication and authorization statement protocol, cases must be considered where one or more of the parties in a legitimate SAML transaction - which operate legitimately within their role for that transaction - attempt to use information gained from a previous transaction maliciously in a subsequent transaction.

SAML threat model

- (Scoping) Assumptions:
  - the local mechanisms that are used to decide whether or not to generate assertions are out of scope. Thus, threats arising from the details of the original login at an authentication authority, for example, are out of scope as well. **If an authority issues a false assertion, then the threats arising from the consumption of that assertion by downstream systems are explicitly out of scope**.
  - **The direct consequence of such a scoping is that the security of a system based on assertions as inputs is only as good as the security of the system used to generate those assertions. When determining what issuers to trust, particularly in cases where the assertions will be used as inputs to authentication or authorization decisions, the risk of security compromises arising from the consumption of false but validly issued assertions is a large one.** ***Trust policies between asserting and relying parties should always be written to include significant consideration of liability and implementations must be provide an audit trail.***

SAML-Specific Security Considerations

- SAML Assertions
  - most concerns arise during communications in the request/response protocol, or during the attempt to use SAML by means of one of the bindings.
  - However, an assertion, once issued, is out of the control of the issuer. This fact has a number of ramifications. For example, the issuer has no control over how long the assertion will persist in the systems of the consumer; nor does the issuer have control over the parties with whom the consumer will share the assertion information. These concerns are over and above concerns about a malicious attacker which can see the contents of assertions that pass over the wire unencrypted (or insufficiently encrypted).

Security considerations for SAML request-response protocol

- Denial of Service
  - The SAML protocol is susceptible to a denial of service (DOS) attack.
  - Handling a SAML request is potentially a very expensive operation, including parsing the request message (typically involving construction of a DOM tree), database/assertion store lookup (potentially on an unindexed key), construction of a response message, and potentially one or more digital signature operations. Thus, the effort required by an attacker generating requests is much lower than the effort needed to handle those requests.

SAML implementations

- **SQLData Systems, Inc** - SQLData SAML Server (http://www.sqldata.com/saml.htm)
- OpenSAML 1.0 - an Open Source SecurityAssertion Markup Language implementation (http://www.opensaml.org/)
- Netegrity (http://www.netegrity.com) recently announced the availability of a free SAML implementation for Java called JSAML that, according to their press release, will be available in October. (http://www.itworld.com/nl/java_sec/09282001/)

SAML implementations

- Shibboleth (http://shibboleth.internet2.edu/)
  - single sign-on software with an emphasis on user privacy, built on the SAML 1.1 specification
  - Use Cases: Delegated trust in portal scenarios (e.g. meta-searching)

XACML - Topics

- Goals
- Approach
- Examples
- Summary

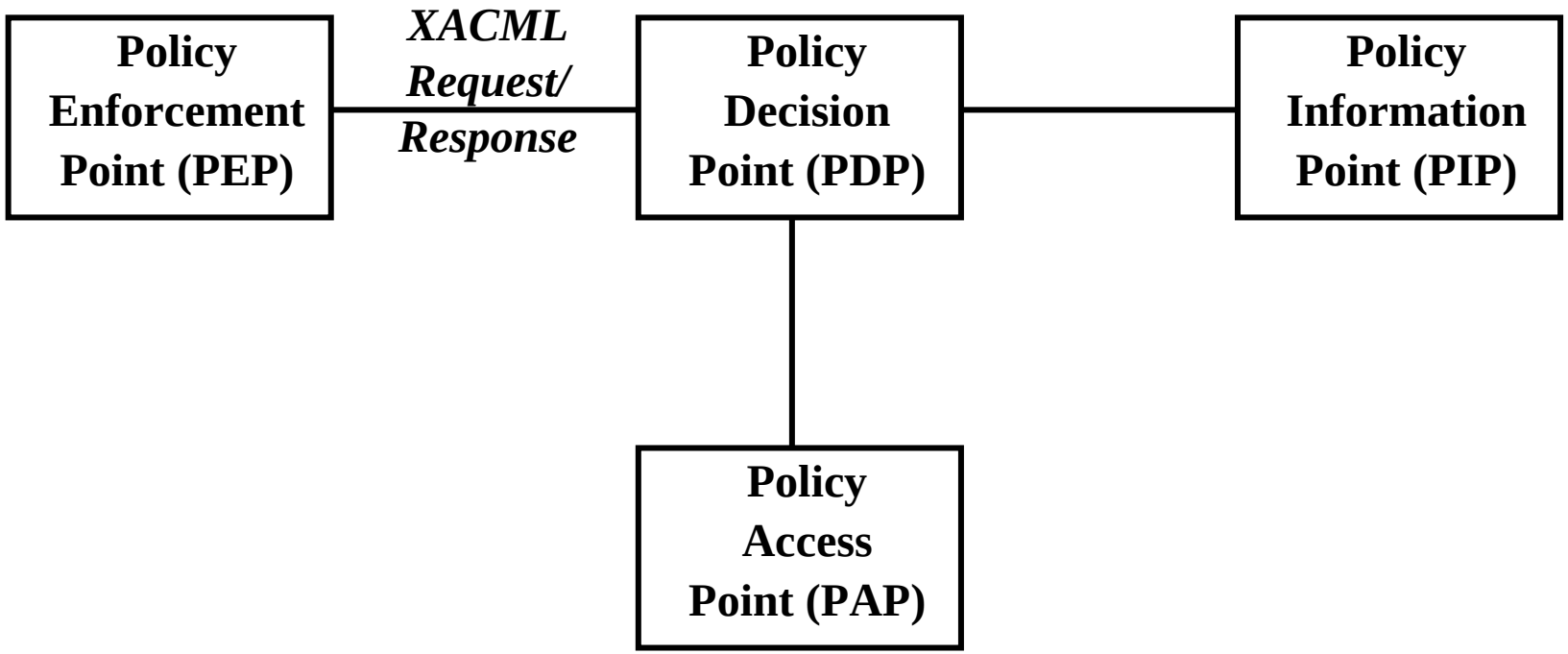
Goals

- Define a core XML schema for representing authorization and entitlement policies
- Target - any object - referenced using XML
- Fine access control grained control
- Access control based on subject and object attributes
- Access control based on the object contents; if the object is not an XML document, the object attributes can be used
- Consistent with and building upon SAML

XACML – Key Aspects

- General-purpose authorization policy model and XML-based specification language
- XACML is independent of SAML specification
- Triple-based policy syntax: <Object, Subject, Action>
- Negative authorization is supported
- Input/output to the XACML policy processor is clearly defined as XACML context data structure
- Input data is referred by XACML-specific attribute designator as well as XPath expression
- Extension points: function, identifier, data type, rule-combining algorithm, policy-combining algorithm, etc.
- A policy consists of multiple rules
- A set of policies is combined by a higher level policy (PolicySet element)

XACML Protocol



XACML Protocol

- When a client makes a resource request upon a server, the PEP is charged with AC
- In order to enforce AC policies, the PEP will formalize the attributes describing the requester at the PIP and delegate the authorization decision to the PDP
- Applicable policies are located in a policy store, managed by the PAP, and evaluated at the PDP, which then returns the authorization decision
- Using this information, the PEP can deliver the appropriate response to the client

XACML Protocol

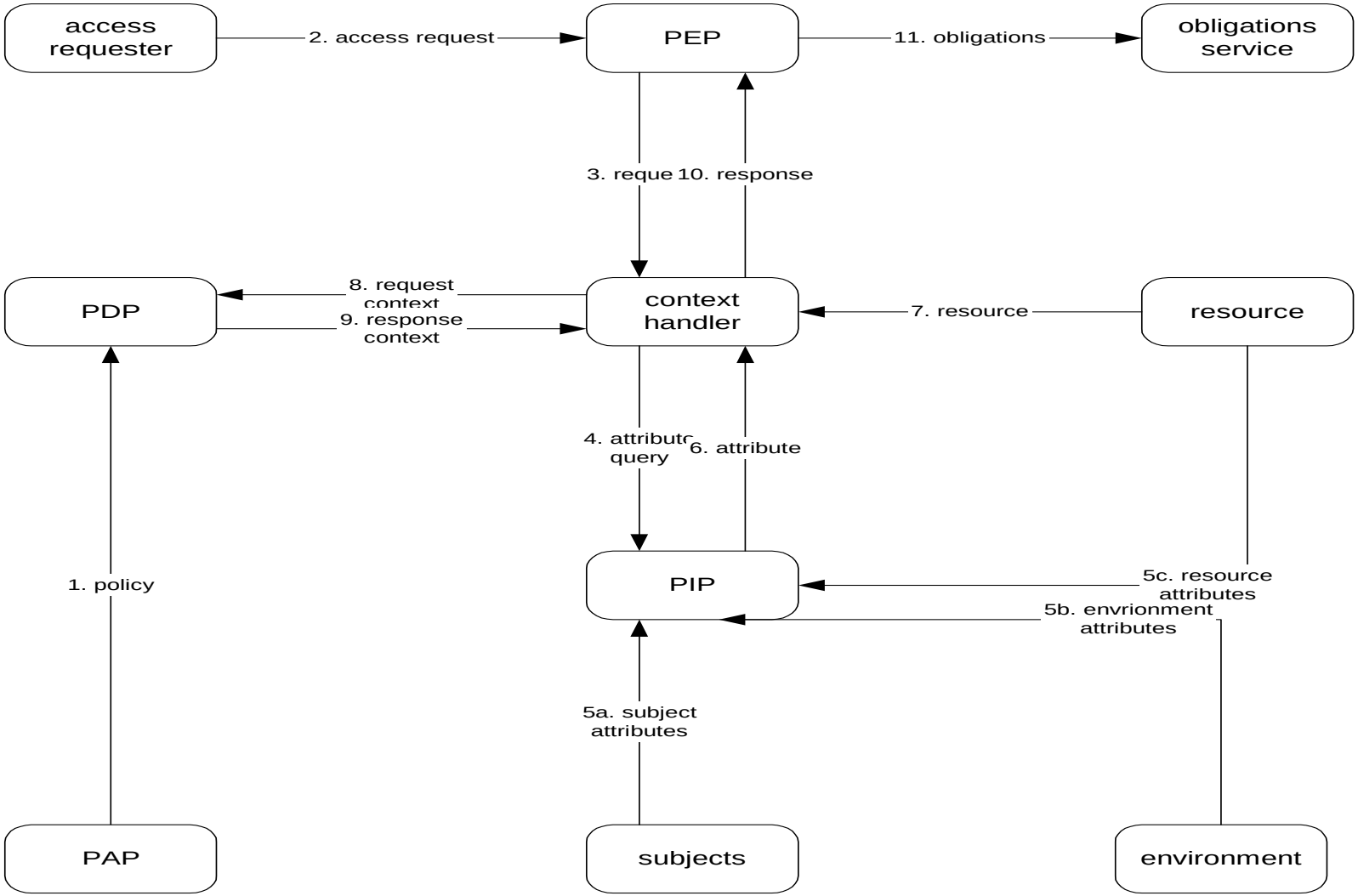
1. The *Policy Administration Point* (PAP) creates security policies and stores these policies in the appropriate repository.
2. The *Policy Enforcement Point* (PEP) performs access control by making decision requests and enforcing authorization decisions.
3. The *Policy Information Point* (PIP) serves as the source of attribute values, or the data required for policy evaluation.
4. The *Policy Decision Point* (PDP) evaluates the applicable policy and renders an authorization decision.

Note: The PEP and PDP might both be contained within the same application, or might be distributed across different servers

XACML Protocol

- XACML Request
  - Subject
  - Object
  - Action
- XACML Response
  - Permit
  - Permit with Obligations
  - Deny
  - NotApplicable (the PDP cannot locate a policy whose target matches the required resource)
  - Indeterminate (an error occurred or some required value was missing)

Data Flow Model



Data Flow Model

1. PAPs write **policies** and **policy sets** and make them available to the PDP. These **policies** or **policy sets** represent the complete policy for a specified **target**
2. The access requester sends a request for access to the PEP
3. The PEP sends the request for **access** to the context handler in its native request format, optionally including **attributes** of the **subjects**, **resource**, **action** and **environment**
4. The **context handler** constructs an XACML request **context** and send it to the PDP
5. The PDP requests any additional **subject**, **resource**, **action**, and **environment attributes** from the context handler
6. The context handler requests the attributes from a PIP
7. The PIP obtains the requested **attributes**
8. The PIP returns the requested **attributes** to the context handler
9. Optionally, the context handler includes the **resource** in the context

Data Flow Model

10. The context handler sends the requested **attributes** and (optionally) the resource to the PDP. The PDP evaluates the **policy**
11. The PDP returns the response **context** (including the **authorization decision**) to the context handler
12. The context handler translates the response context to the native response format of the PEP. The context handler returns the response to the PEP
13. The PEP fulfills the **obligations**
14. (Not shown) If **access** is permitted, then the PEP permits **access** to the **resource**; otherwise, it denies **access**

XACML Schemas

Request Schema

Request

- Subject
- Resource
- Action

Policy Schema

PolicySet (Combining Alg)

- Policy* (Combining Alg)
  - Rule* (Effect)
    - Target
      - Subject*
      - Resource*
      - Action*
      - Environment
    - Effect
    - Condition
    - Obligation*

Response Schema

Response

- Decision
- Obligation*

XACML Schemas

Request Schema

Request

- Subject
- Resource
- Action

Policy Schema

PolicySet (Combining Alg)

- Policy* (Combining Alg)
  - Rule* (Effect)
    - Subject*
    - Resource*
    - Action
    - Condition*
    - Obligation*

Response Schema

Response

- Decision
- Obligation*

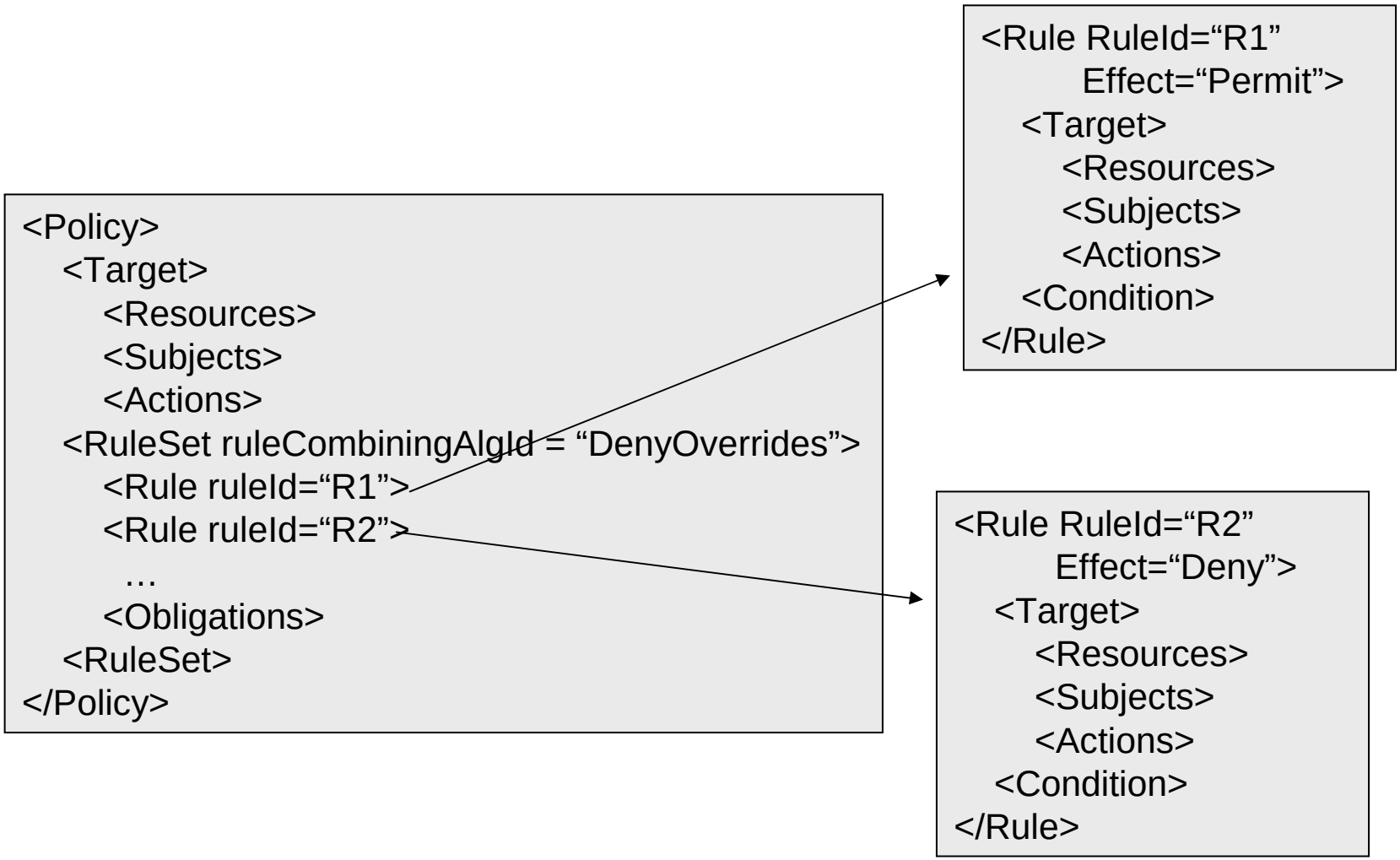
Policies and PolicySet

- The key top-level element is the <PolicySet> which aggregates other <PolicySet> elements or <Policy> elements
- The <Policy> element is composed principally of <Target>, <RuleSet> and <Obligation> elements and is evaluated at the PDP to yield and access decision.
- Since multiple policies may be found applicable to an access decision, (and since a single policy can contain multiple Rules) Combining Algorithms are used to reconcile multiple outcomes into a single decision
- The <Target> element is used to associate a requested resource with an applicable Policy. It contains conditions that the requesting Subject, Resource, or Action must meet for a Policy Set, Policy, or Rule to be applicable to the resource.
- The Target includes a build-in scheme for efficient indexing/lookup of Policies.
- Rules provide the conditions which test the relevant attributes within a Policy. Any number of Rule elements may be used each of which generates a true or false outcome. Combining these outcomes yields a single decision for the Policy, which may be "Permit", "Deny", "Indeterminate", or a "NotApplicable" decision.

Policies and Policy Sets

- Policy
  - Smallest element PDP can evaluate
  - Contains: Description, Defaults, Target, Rules, Obligations, Rule Combining Algorithm
- Policy Set
  - Allows Policies and Policy Sets to be combined
  - Use not required
  - Contains: Description, Defaults, Target, Policies, Policy Sets, Policy References, Policy Set References, Obligations, Policy Combining Algorithm
- Combining Algorithms: Deny-overrides, Permit-overrides, First-applicable, Only-one-applicable

Overview of the Policy Element



Combining Algorithms

- Policy & Rule Combining algorithms

Permit Overrides:

If a single rule permits a request, irrespective of the other rules, the result of the PDP is Permit

Deny Overrides:

If a single rule denies a request, irrespective of the other rules, the result of the PDP is deny.

First Applicable:

The first applicable rule that satisfies the request is the result of the PDP

Only-one-applicable:

If there are two rules with different effects for the same request, the result is indeterminate

Rules

- Smallest unit of administration, cannot be evaluated alone
- Elements
  - Description – documentation
  - Target – select applicable rules
  - Condition – boolean decision function
  - Effect – either “Permit” or “Deny”
- Results
  - If condition is true, return Effect value
  - If not, return NotApplicable
  - If error or missing data return Indeterminate
    - Plus status code

Target

- Designed to efficiently find the policies that apply to a request
- Makes it feasible to have very complex Conditions
- Attributes of Subjects, Resources and Actions
- Matches against value, using match function
  - Regular expression
  - RFC822 (email) name
  - X.500 name
  - User defined
- Attributes specified by Id or XPath expression
- Normally use Subject or Resource, not both

Rule Element

- The main components of the <rule> element are:
  - a <target>
    - the <target> element consists of
      - a set of <resource> elements
      - a set of <action> elements
      - an environment
    - the <target> element may be absent from a <rule>. In this case the <target> of the rule is the same as that of the parent <policy> element
  - an <effect>
    - Two values are allowed: “Permit” and “Deny”
  - a <condition>

Policy Element

- The main components of a <policy> element are:
  - a <target> element
    - the <target> element consists of
      - a set of <resource> elements
      - a set of <action> elements
      - an environment
    - the <target> element may be declared explicitly or may be calculated; two possible approaches:
      - Make the union of all the target elements in the inner rules
      - Make the intersection of all the target elements in the inner rules
  - a rule-combining algorithm-identifier
  - a set of <rule> elements
  - obligations

PolicySet Element

- The main components of a <policyset> element are:
  - a <target>
  - a policy-combining algorithm-identifier
  - a set of <policy> elements
  - obligations

A Policy Example

- The Policy applies to requests for the server called “SampleServer”
- The Policy has a Rule with a Target that requires an action of "login" and a Condition that applies only if the Subject is trying to log in between 9am and 5pm.
- Note that this example can be extended to include other Rules for different actions.
- If the first Rule does not apply, then a default Rule is used that always returns Deny (Rules are evaluated in order).

A Policy Example

```
<Policy PolicyId="SamplePolicy"
    RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
    algorithm:permit-overrides">
    <!-- This Policy only applies to requests on the SampleServer -->
    <Target>
      <Subjects> <AnySubject/> </Subjects>
      <Resources>
          <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
    equal">
          <AttributeValue
    DataType="http://www.w3.org/2001/XMLSchema#string">SampleServer
              </AttributeValue>
          <ResourceAttributeDesignator
    DataType="http://www.w3.org/2001/XMLSchema#string"
          AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
    </ResourceMatch>
      </Resources>
      <Actions> <AnyAction/> </Actions>
    </Target>
```

A Policy Example

```
<!-- Rule to see if we should allow the Subject to login -->
<Rule RuleId="LoginRule" Effect="Permit">
<!-- Only use this Rule if the action is login -->
    <Target>
      <Subjects> <AnySubject/> </Subjects>
     <Resources> <AnyResource/> </Resources>
     <Actions>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
    equal">
        <AttributeValue

    DataType="http://www.w3.org/2001/XMLSchema#string">login</AttributeV
    alue>
         <ActionAttributeDesignator
    DataType=http://www.w3.org/2001/XMLSchema#string
        AttributeId="ServerAction"/>
       </ActionMatch>
     </Actions>
   </Target>
```

A Policy Example

```
<!-- Only allow logins from 9am to 5pm -->
<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-
   equal"
     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
          <EnvironmentAttributeSelector
          DataType="http://www.w3.org/2001/XMLSchema#time"
          AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
   </Apply>
  <AttributeValue
   DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValu
   e> </Apply>
 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal"
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
          <EnvironmentAttributeSelector
          DataType="http://www.w3.org/2001/XMLSchema#time"
   AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
   </Apply>
   <AttributeValue
   DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00</AttributeValu
   e> </Apply>
</Condition>
```

Condition

- Boolean function to decide if Effect applies
- Inputs come from Request Context
- Values can be primitive, complex or bags
- Can be specified by id or XPath expression
- Fourteen primitive types
- Rich array of typed functions defined
- Functions for dealing with bags
- Order of evaluation unspecified
- Allowed to quit when result is known
- Side effects not permitted

Functions

- Equality predicates
- Arithmetic functions
- String conversion functions
- Numeric type conversion functions
- Logical functions
- Arithmetic comparison functions
- Date and time arithmetic functions
- Non-numeric comparison functions
- Bag functions
- Set functions
- Higher-order bag functions
- Special match functions
- XPath-based functions
- Extension functions and primitive types

Request and Response Context

- Request Context
  - Attributes of:
    - Subjects – requester, intermediary, recipient, etc.
    - Resource – name, can be hierarchical
    - Resource Content – specific to resource type, e.g. XML document
    - Action – e.g. Read
    - Environment – other, e.g. time of request
- Response Context
  - Resource ID
  - Decision
  - Status (error values)
  - Obligations

XACML History

- First Meeting – 21 May 2001
- Requirements from: Healthcare, DRM, Registry, Financial, Online Web, XML Docs, Fed Gov, Workflow, Java, Policy Analysis, WebDAV
- XACML 1.0 - OASIS Standard – 6 February 2003
- XACML 1.1 – Committee Specification – 7 August 2003
- XACML 2.0 – In progress – complete summer 2004