Technical Lesson 3.1.4

Referencing Resource Content

Goals:

1. Understand how to use the **<AttributeSelector>** element.
2. Understand how a Policy can use the content of resources in the evaluation process.

Summary:

This Lesson introduces the use of the **<AttributeSelector>** element. You will be asked to inspect and analyze policies using this element, and make comparisons with policies that only use *attribute-designators*. You will confirm the analysis by evaluating the policies against requests. Finally, you will be challenged to edit a policy and a request to achieve specific results.

Steps:

3.1.4.1 Inspect Permit-Policy.xml and Request-1.xml

Confirm that this Permit-Policy and Request-1 are the same as the Permit-Policy and Request-1 from Lesson 3.1.1.

3.1.4.2 Evaluate Permit-Policy against Request-1

Recall from Lesson 3.1.1 that the **<Decision>** should be "Permit". Confirm that this is the case.

3.1.4.3 Inspect Selector-Policy.xml

Selector-Policy is semantically the same as Permit-Policy, with two significant syntactical differences. Recall from Lesson 3.1.1 that a *match-predicate* consist of three parts:

- A **MatchId**
- An **<AttributeValue>**
- An *attribute-reference* which can be an *attribute-designator* or an **<AttributeSelector>**
    - The name of *attribute-designator* elements are dependent on which *class* the *attribute-designator* is in. **<Subjects>** contain **<SubjectAttributeDesignator>** elements and so on.

The **<SubjectMatch>** *match-predicate* of Selector-Policy uses an **<AttributeSelector>** *attribute-reference* (Lines 18 – 19), while the **<SubjectMatch>** of Permit-Policy uses an *attribute-designator* (**<SubjectAttributeDesignator>**) *attribute-reference* (Lines 17 – 18). The particular **<AttributeSelector>** in Selector-Policy causes the exact same semantic effect as the

<SubjectAttributeDesignator> in Permit-Policy: it causes the PDP, when evaluating the policy against a request, to retrieve the values of all **<AttributeValue>** elements (as a **bag** of values) of the GFIPM Security Clearance Level Code Subject attribute of the request.

An **<AttributeSelector>** contains a **DataType** and a **RequestContextPath**. The value of a **RequestContextPath** must be an XPath expression into the request context[1]. The PDP will retrieve the set of nodes[2] referenced by the **RequestContextPath** as a **bag** of values. If no nodes are found, then the PDP returns an empty **bag**[3].

Beware that XACML defines one XML namespace for policies and a separate namespace for the XACML context. Since a **RequestContextPath** is an XPath expressions into the request context, any policy that uses an **<AttributeSelector>** must declare the XACML context namespace. This is done in Selector-Policy on Line 4; a prefix of “ctx” is used to represent the context namespace. On Line 19, the “ctx” prefix is used in the XPath expression.

3.1.4.4 Evaluate Selector-Policy against Request-1

Confirm that the **<Decision>** for “Resource-1” is “Permit”.

3.1.4.5 Inspect ArrestRecord.xsd

This file is in the “$POLICY_GUIDE/arrest_record_simple/” directory. It contains an XML Schema[4] for an **<ArrestRecord>** element. We will use this schema to represent a set of Arrest Records for which we want to protect access.

The schema defines an **<ArrestRecord>** element that contains seven sub-elements:

- **<Id>** - the identifier of the record.
- **<SubjectId>** - the identifier of the individual who was arrested.
- **<Jurisdiction>** - the jurisdiction in which the arrest occurred.
- **<Date>** - the date at which the arrest occurred.
- **<ArrestingOfficerId>** - the identifier of the arresting officer.
- **<ArrestingOfficerAgencyName>** - the name of the agency that employs the arresting officer.
- **<ArrestingOfficerEmailAddress>** - the email address of the arresting officer.

[1] The XACML “context” is the XML structures of requests and responses.
[2] A “node” is a term used in the context of XPath that means a part of an XML document.
[3] The optional **MustBePresent** property of *attribute-references* changes this behavior.
[4] The IEPD schema used here is technically not a genuine IEPD; however, the schema is NIEM IEPD-conformant and provides a close approximation of a genuine IEPD.

An arrest record articulates that an officer arrested some individual on a particular date within a particular jurisdiction. Valid values for jurisdiction are defined by the GFIPM Jurisdiction Code Set[5].

3.1.4.6 Inspect Record-1.xml

Confirm that this XML document conforms to ArrestRecord.xsd[6]. Record-1 states that Officer-1 arrested Subject-1 in Georgia on Valentine's Day 2012.

3.1.4.7 Inspect Content-Request-1.xml

This request shows an example of how XML content can be included in a request. The **<ResourceContent>** element (Lines 8 – 16) contains the content of the Record-1 Arrest Record (Lines 9 – 15). Notice the declaration of the Arrest Record namespace on Line 10 and the use of the "ar" prefix throughout the content of the Arrest Record.

Policies must use an **<AttributeSelector>** to retrieve values from a **<ResourceContent>** element in a request.

3.1.4.8 Inspect Content-Policy-1.xml

This policy will evaluate to "Permit" for requests that contain an Arrest Record with a Jurisdiction value of "GA".

The **<Target>** (Lines 16 – 27) contains a single **<ResourceMatch>** *match-predicate* (Lines 19 – 24) that uses an **<AttributeSelector>** (Lines 21 – 23). The **RequestContextPath** (Line 23) expression points to the value of the **<Jurisdiction>** element in an **<ArrestRecord>**.

Notice the use of the "ar" namespace prefix in the XPath expression and the declaration of the Arrest Record namespace on Line 5. When using the SunXACML library, XPath expressions in **RequestContextPath** XML-attributes must be XML namespace qualified.

This policy contains a single "Permit" **<Rule>** that has an empty **<Target>**.

3.1.4.9 Evaluate Content-Policy-1 against Content-Request-1

[5] The GFIPM Jurisdiction code set is available at http://gfipm.net/standards/metadata/2.0/codesets/GFIPMJurisdictionCode.html.
[6] This can be done by using an XML Schema validator to validate Record-1.xml against ArrestRecord.xsd.

First, let's manually determine what the result should be. If the single **<ResourceMatch>** of the policy evaluates to true, then the policy should evaluate to "Permit", otherwise the policy should evaluate to "NotApplicable".

The **<ResourceMatch>** will be true for requests that include an **<ArrestRecord>** that has a **<Jurisdiction>** value of "GA". Content-Request-1 has such an **<ArrestRecord>**, therefore Content-Policy-1 should evaluate to "Permit".

Now, execute SimplePDP with Content-Request-1.xml and Content-Policy-1.xml, and output the results to Content-Request-1_Content-Policy-1_Response.xml. Confirm that the **<Decision>** for "Resource-1" is "Permit".

3.1.4.10 Challenge: Add match-predicates to Content-Policy-1

Create a copy of Content-Policy-1 and call the new file: "Content-Policy-2.xml". Open Content-Policy-2.xml. On Line 5, change the end of the **PolicyId** to read "Content-Policy-2".

In the existing **<Resource>** *instance* (Lines 18 – 25), create a new **<ResourceMatch>** that articulates this predicate: "the Subject Id of the Arrest Record equals 'Subject-1'."

Notice that the subject of the Arrest Record is handled in the **<Resources>** *class* because it is a part of the resource content and is not the subject of the request.

Create the **<Subjects>** *class* (currently non-existent) in the **<Target>** of the policy, and include one **<Subject>** *instance*. In this **<Subject>**, create a **<SubjectMatch>** that articulates this predicate: "The request Subject is a Sworn Law Enforcement Officer." Use the GFIPM Sworn Law Enforcement Officer Indicator attribute identifier[7].

A solution to this Challenge is in Content-Policy-2-Solution.xml.

3.1.4.11 Evaluate Content-Policy-2 against Content-Request-1

Evaluate your Content-Policy-2 against Content-Request-1. Confirm that the **<Decision>** for "Resource-1" is "NotApplicable". This should be because the **<SubjectMatch>** you created in Content-Policy-2 should evaluate to false; Content-Request-1 is silent on Subject attributes. Recall that all four *classes* must match a request in order for the parent **<Target>** to match the request.

[7] Details on the GFIPM Sworn Law Enforcement Officer Indicator attribute are at http://gfipm.net/standards/metadata/2.0/user/SwornLawEnforcementOfficerIndicator.html.

3.1.4.12 Challenge: Create a request that is applicable to Content-Policy-2

Create a copy of Content-Request-1 and call the new file: “Content-Request-2”. Edit Content-Request-2 to make it applicable to Content-Policy-2.

A solution to this Challenge is in Content-Request-2-Solution.xml.

3.1.4.13 Evaluate Content-Policy-2 against Content-Request-2

Evaluate your Content-Policy-2 against Content-Request-2. Confirm that the **<Decision>** for “Resource-1” is “Permit”.

A Note about Notation

XML elements, for XACML and data files, are written as they appear in XML documents, and are indicated in boldface text. For example: **<Policy>**.

XML attributes, for XACML and data files, are written as they appear in XML documents, and are indicated in boldface text. For example: **PolicyId**.

Values of XACML and data elements appear in double quotes. For example: “Permit”.

We introduce some terms to serve as labels for certain groups of policy elements; these terms are used to enable discussions about groups of elements as a whole. These terms appear in italics. For example: *class*.

We use labels to refer to files, directories, and data items that exist in the accompanying virtual machine. These labels are used in the style of Linux environment variables – they begin with a dollar sign ($) which is followed by the label in all caps. For example: the label $POLICY_GUIDE refers to the following path on the virtual machine, “/home/guide/policy-guide”.