Technical Lesson 3.1.6

Aggregating Multiple Rules

Goals:

1. Understand how to aggregate multiple rules into a single policy.
2. Understand the potential for conflicts.
3. Understand how rule combining algorithms are used.

Summary:

In this Lesson, you will inspect, analyze, manipulate, and evaluate policies that have multiple rules. You will learn about conflicts among rules and how rule-combining algorithms resolve those conflicts. Also, you will be challenged with authoring a policy that expresses a source policy with multiple rules.

Steps:

3.1.6.1 Inspect Policy-1.xml

This policy has an empty **<Target>** and contains two rules. The first rule (Lines 16 – 35), "Rule-1", has an **Effect** of "Permit". The second rule (Lines 37 – 65), "Rule-2", has an **Effect** of "Deny". Rule-1 can be read: "Officers can perform any Action in any Environment on Arrest Records for which they are the arresting officer." Rule-2 can be read: "Arrest Records in the 'MD' Jurisdiction cannot be deleted by any Subject in any Environment."

The two rules have conflicting **Effect** values. During evaluation against a request, if both rules are applicable to the request, the policy will evaluate to "Deny" due to its rule-combining algorithm.

The rule-combining algorithm of the policy is "deny-overrides" (see Line 6). With "deny-overrides", if any rule evaluates to "Deny", then the policy will evaluate to "Deny". If no rule evaluates to "Deny", but at least one rule evaluates to "Permit", then the policy will evaluate to "Permit". Otherwise, the policy will evaluate to "NotApplicable".

Along with "deny-overrides", main rule-combining algorithms available in XACML are "permit-overrides" and "first-applicable"[1]. The "permit-overrides" algorithm can be considered the inverse of "deny-overrides": "Permit" decisions take precedence over "Deny" decisions. With the "first-applicable" algorithm, the rules are evaluated in the order as they appear in the

[1] The complete list and semantic definitions of all standard rule combining algorithms is in Appendix C of the XACML 2.0 Specification.

policy; the policy evaluates to the **Effect** of the first rule that is applicable to the request, or "NotApplicable" if no rules are applicable.[2]

3.1.6.2 Inspect Request-1.xml

Request-1 is the articulation of a request by Officer-1 to delete Resource-1 which is an Arrest Record. Officer-1 is the arresting officer and the arrest Jurisdiction is "VA".

3.1.6.3 Evaluate Policy-1 against Request-1

First, let's manually determine what the result should be. Since the policy uses the "deny-overrides" combining algorithm, we should check Rule-2 (the "Deny" rule) first. Rule-2 is not applicable to the request since the Jurisdiction in the request is "VA" and not "MD".

Now, let's consider Rule-1. Rule-1 is applicable to the request since Officer-1 is attempting access on a record of which Officer-1 is the arresting officer. Therefore, the policy should evaluate to "Permit".

Confirm that the **<Decision>** of Resource-1 is "Permit".

3.1.6.4 Inspect Request-2.xml

Request-2 is the articulation of a request by Officer-2 to delete Resource-2, which is an Arrest Record. Officer-2 is the arresting officer and the Jurisdiction is "MD".

3.1.6.5 Evaluate Policy-1 against Request-2

First, let's manually determine what the result should be. We'll check Rule-2 first. Rule-2 should be applicable to the request since the Jurisdiction in the request is "MD". Since a "Deny" rule is applicable, and since the rule-combining algorithm is "deny-overrides", there is no need to check Rule-1. The policy should evaluate to "Deny".

Confirm that the **<Decision>** of Resource-2 is "Deny".

3.1.6.6 Inspect Request-3.xml

Request-3 is the articulation of a request by Officer-3 to read Resource-3, which is an Arrest Record. Officer-4 is the arresting officer and the Jurisdiction is "MD".

[2] These are simplified descriptions of the semantics of "deny-overrides", "permit-overrides", and "first-applicable". These algorithms also handle cases where a rule evaluates to "Indeterminate".

3.1.6.7 Evaluate Policy-1 against Request-3

First, let's manually determine what the result should be. We'll check Rule-2 first. Rule-2 should not be applicable to Request-3 since Rule-2 applies to the delete Action and Request-3 seeks a read Action.

Now, let's consider Rule-1. Rule-1 should not be applicable to the request since the request Subject, Officer-3, does not match the arrest record's OfficerID, Officer-4. Therefore, the policy should evaluate to "NotApplicable".

Confirm that the **<Decision>** for Resource-3 is "NotApplicable".

3.1.6.8 Challenge: Create a new policy with multiple rules

The **<Description>** of Policy-1 (Lines 8 – 12) states: "Officers can perform any Action on Arrest Records for which they are the arresting officer. However, under no circumstances can records in the 'MD' Jurisdiction be deleted." Your Challenge is to create a new policy with a slightly different articulation: "Officers can perform any Action on Arrest Records for which they are the arresting officer. However, under no circumstances can records in the 'MD' Jurisdiction be deleted, except by holders of a Top Secret Clearance. Holders of a Top Secret Clearance can perform any Action on any Record in any Environment."

Save your new policy in a file called "Policy-2.xml". There are several possible solutions to this Challenge. One solution is provided in Policy-2-Solution.xml.

3.1.6.9 Inspect Policy-2-Solution.xml

Let's compare Policy-2-Solution to Policy-1. The rule-combining algorithm was changed to "first-applicable". A new Rule-1 provides total access to request Subjects with a Top Secret Security Clearance Level Code. Rule-2 stayed the same. Rule-1 from Policy-1 became Rule-3 in Policy-2-Solution.

The Description of Rule-1 of Policy-2-Solution (Lines 20 – 23) states: "Holders of a Top Secret Clearance can perform any Action on any Record in any Environment." When evaluating this policy against a request, the PDP will first evaluate Rule-1. If Rule-1 applies to a request, then the "first-applicable" rule-combining algorithm tells the PDP to proceed no further and to apply the Effect of Rule-1: "Permit". If Rule-1 is not applicable to the request, then the PDP will evaluate Rule-2. If Rule-2 is not applicable to the request, then the PDP will evaluate Rule-3. If Rule-3 is not applicable, then the policy will evaluate to "NotApplicable".

3.1.6.10 Inspect Request-4.xml

Request-4 can be articulated as follows: “Officer-4, who has a Top Secret Clearance, is attempting to delete Resource-4, which is an Arrest Record. Officer-4 is the arresting officer and the Jurisdiction is ‘MD’.”

3.1.6.11 Evaluate Policy-2 against Request-4

Use Request-4 to test your Policy-2 (and Policy-2-Solution). Confirm that the **<Decision>** for Resource-4 evaluates to “Permit”, since the request matches Rule-1.

3.1.6.12 Inspect Request-5.xml

Request-5 can be articulated as follows: “Officer-5, who has a “Secret” Clearance, is attempting to delete Resource-5, which is an Arrest Record. Officer-5 is the arresting officer and the Jurisdiction is ‘MD’.”

3.1.6.13 Evaluate Policy-2 against Request-5

Use Request-5 to test your Policy-2 (and Policy-2-Solution). Rule-1 should not be applicable to the request since Officer-5 does not have a “Top Secret” Clearance. Rule-2 should be applicable since the request is an attempt to delete an Arrest Record in the “MD” Jurisdiction. Therefore, Policy-2 should evaluate to “Deny”.

Confirm that the **<Decision>** for Resource-5 is “Deny”.

3.1.6.14 Inspect Request-6.xml

Request-6 can be articulated as follows: “Officer-6, who has a Secret Clearance, is attempting to delete Resource-6, which is an Arrest Record. Officer-6 is the arresting officer and the Jurisdiction is ‘VA’.”

3.1.6.15 Evaluate Policy-2 against Request 6

Use Request-6 to test your Policy-2 (and Policy-2-Solution). Rule-1 should not be applicable to the request since Officer-6 does not have a Top Secret Clearance. Rule-2 should not be applicable since the Jurisdiction of the record is not “MD”. Rule-3 should be applicable since Officer-6 is both the Subject of the request and the arresting officer on the record. Therefore, Policy-2 should evaluate to “Permit”.

Confirm that the **<Decision>** for Resource-6 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”.