Policy Interoperability Through Rules Interchange Format

Jie Bao[1], Lalana Kagal[2], and Vladimir Kolovski[3]

[1] Rensselaer Polytechinc Institute
Troy, NY 12180
baojie@cs.rpi.edu
[2] MIT CSAIL
Cambridge, MA 02139
lkagal@csail.mit.edu
[3] Semantic Technologies Group, Oracle Corporation
Nashua, NH 03062
vladimir.kolovski@oracle.com

Abstract. With policy management being a popular mechanism for providing flexible security, the number of policy languages being proposed is constantly increasing. The use of disparate policy languages makes collaboration between and integration over applications and domains difficult and requires prior negotiation and agreement between them. These problems could be eased by the adoption of a standard policy language. Users, however, should not be forced to conform to a single language as that will reduce their flexibility and autonomy. Rule Interchange Format (RIF) is a W3C recommendation that allows the exchange of rules between rule systems. We propose creating a policy interchange language grounded in RIF. This policy interchange language will allow systems to use their own policy language while still collaborating securely with those that do not. We plan to study several policy languages and capture their main functionality in RIF with an aim to identifying their common subset. Our intuition leads us to believe that a generalized form of this common subset, expressed in RIF, will act as a policy interlingua. eXtensible Access Control Markup Language (XACML) is an OASIS standard language for the specification of access control policies and has been deployed in many Web-based systems. As a first step, we show how the semantics of XACML can be expressed in RIF. In this paper, we present our translation between XACML and RIF that allows XACML and non-XACML systems to collaborate while maintaining their security policies.

1 Introduction

XACML [3]
contributions?

- 1
- 2
- 3

2 Motivating Example

show a use case of rule interoperability

3 XACML

XACML 3.0

Some features not covered: obligation, delegation, operation parameters (e.g., the `CombinedDecision` attibute in `Request`)

Also not covered in the syntax: annotations, versioning, defaults, status etc.

Simplified abstract syntax of XACML 3.0[4]:

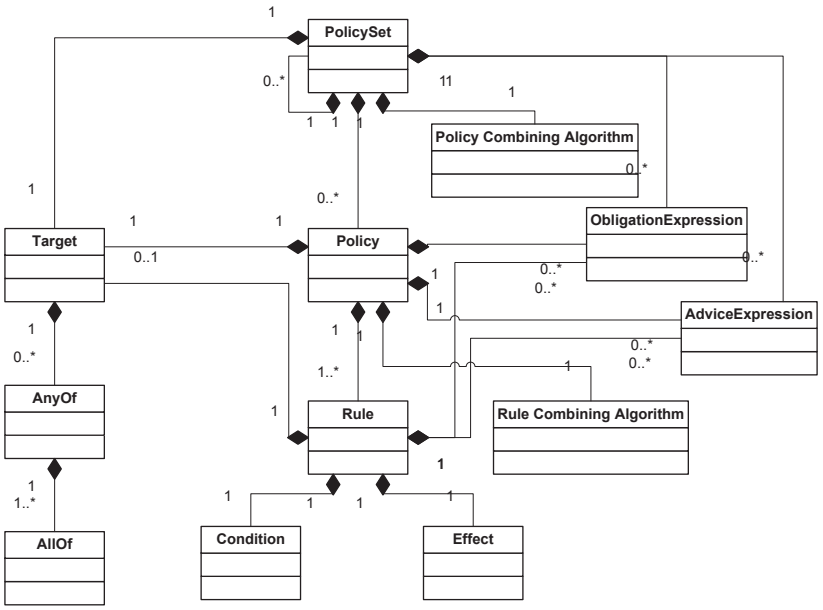

Fig. 1. XACML language model (Figure 3 in [3])

More about functions, expressions and variables

Request matching etc

Decision - 4 possible result of a policy evaluation: "Permit", "Deny", "Indeterminate" (unable to evaluate, e.g., due to missing attributes or), "NotApplicable" (no policy can be applied).

[4] We use the W3C version of the EBNF grammar: `http://www.w3.org/TR/REC-xml/#sec-notation`. *Terminals* are in the Italic font.

```
        PolicySet ::= uriId PolicyCombiningAlg Target (PolicySet|Policy)*
           Policy ::= uriId RuleCombiningAlg Target (VariableDefinition|Rule)*
  RuleCombiningAlg ::= "deny-overrides"| "permit-overrides" | "first-applicable" |
                       "ordered-deny-overrides" | "ordered-permit-overrides" |
                       "deny-unless-permit" | "permit-unless-deny"
PolicyCombiningAlg ::= RuleCombiningAlg | "only-one-applicable"
              Rule ::= strId Target? Condition? Effect
            Target ::= (AnyOf)* /* disjunction */
             AnyOf ::= (AllOf)+ /* conjunction */
             AllOf ::= (Match)+
             Match ::= uriId AttributeValue (AttributeDesignator|AttributeSelector)
            Effect ::= "Permit" | "Deny"
         Condition ::= Expression /* must be a Boolean function*/
```

Fig. 2. Policy and Rule Structure in XACML

```
        Expression ::= Apply | AttributeSelector | AttributeValue |
                       Function | VariableReference | AttributeDesignator
             Apply ::= Function Expression? /* function call*/
  AttributeSelector ::= Category Path DataType MustBePresent
AttributeDesignator ::= Category Attribute DataType MustBePresent
           Function ::= uriId /* e.g., integer-add and string-equal */
     AttributeValue ::= Literal DataType
           DataType ::= "string" | "boolean" | "integer" ...
      MustBePresent ::= "true" | "false"
 VariableDefinition ::= strId Expression
  VariableReference ::= strId
```

Fig. 3. Expressions in XACML

```
   Request ::= Attributes+
Attributes ::= Category Content* Attribute*
 Attribute ::= uriId AttributeValue+
  Category ::= "subject" | "resource" | "action" | "environment"
   Content ::= XMLLiteral
     strId ::= string
     uriId ::= uri
```

Fig. 4. Other schema elements in XACML

4 Representing XACML Policies in RIF

RIF-PRD [1]

4.1 Mapping Policy and Rule Structure

4.2 Mapping Rules

4.3 Request Matching

4.4 Mapping Rule Combining Algorithms

4.5 Correctness of the Mapping

5 Related Work

Other XACML semantics, [2]

Cite "the formal semantics of XACML" by Polar - why it is not a good one for us

Argue about the generality of our approach - other languages that have a logic program translation can be mapped to RIF in the similar way

6 Summary and Future Work

Acknowledgements

This work was carried out with funding from ??

References

1. C. de Sainte Marie, G. Hallmark, and A. Paschke. RIF Production Rule Dialect. Recommendation REC-rif-prd-20100622, World Wide Web Consortium, June 2010.
2. V. Kolovski, J. Hendler, and B. Parsia. Analyzing web access control policies. In *WWW*, pages 677–686, 2007.
3. E. Rissanen. eXtensible Access Control Markup Language (XACML) Version 3.0. Committee Draft xacml-3.0-core-spec-cd-03-en, OASIS, March 2010.