



Key Differences Between XACML and EPAL

New Challenges for Access Control 2005

Anne Anderson
Staff Engineer
Sun Microsystems Labs
Burlington, MA, USA
Anne.Anderson@sun.com

Copyright ©
2005 Sun
Microsystems,
Inc. All rights
reserved.



Outline

- Overview
 - XACML
 - EPAL
- Choice of rules to evaluate
- Policy vocabulary
- Hierarchical Request objects
 - XML document resources
- Subjects
- Conclusion

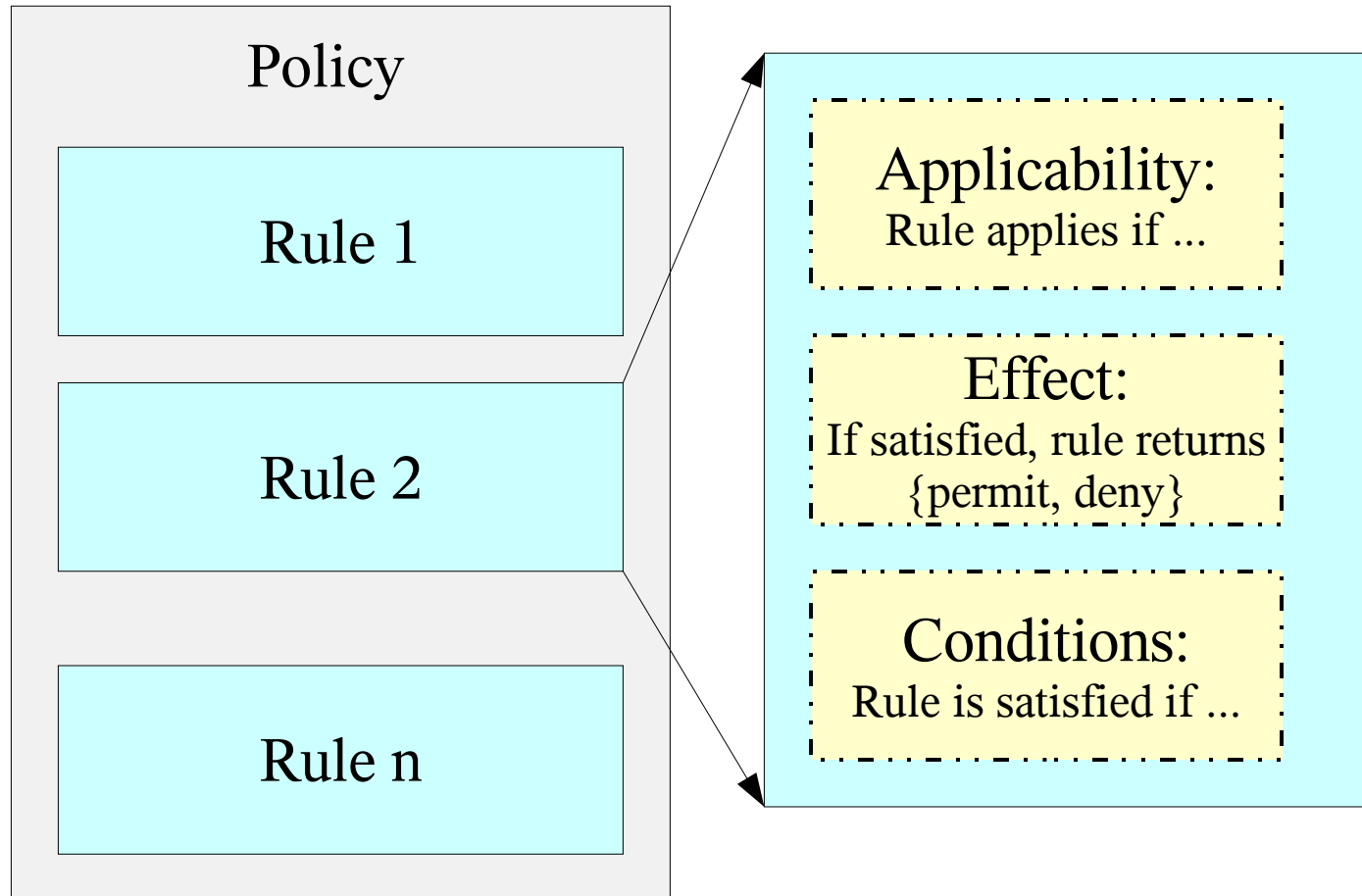
XACML Overview

- “eXtensible Access Control Markup Language”
- Version 1.0 OASIS Standard, February 2003
- Version 2.0 OASIS Standard, February 2005
 - Includes “Privacy profile of XACML”
- Publicly available (C++, C#) and open source (Java) implementations

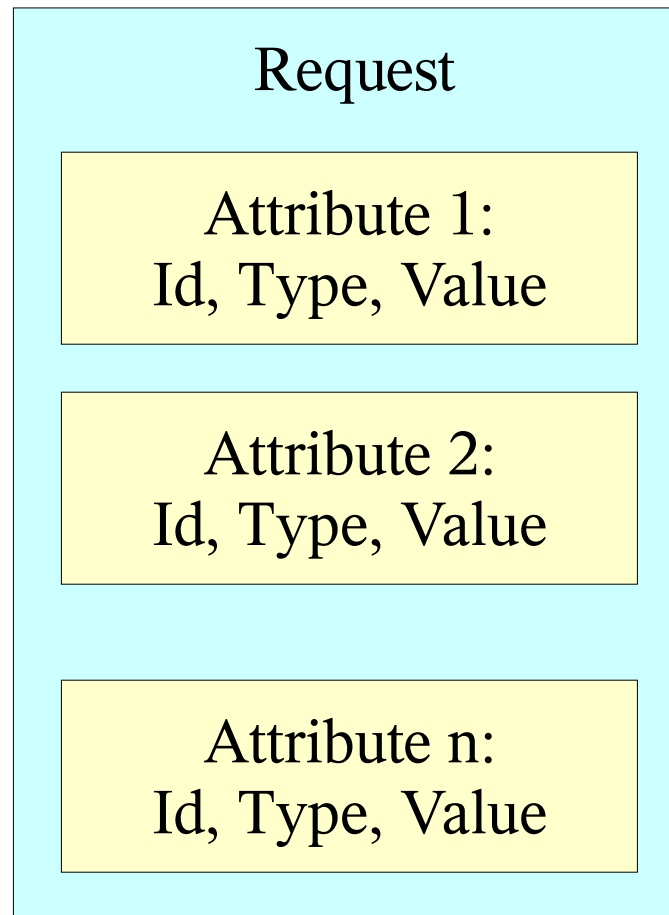
EPAL Overview

- “Enterprise Privacy Authorization Language”
- IBM proprietary specification
- Submitted to W3C 10 November 2003; no action
- EPAL 1.1 used XACML explicitly
- EPAL 1.2 uses a lot of XACML (attribute and target concepts, rules, conditions, functions, datatypes, obligations)

Common Framework: Policy



Common Framework:Request



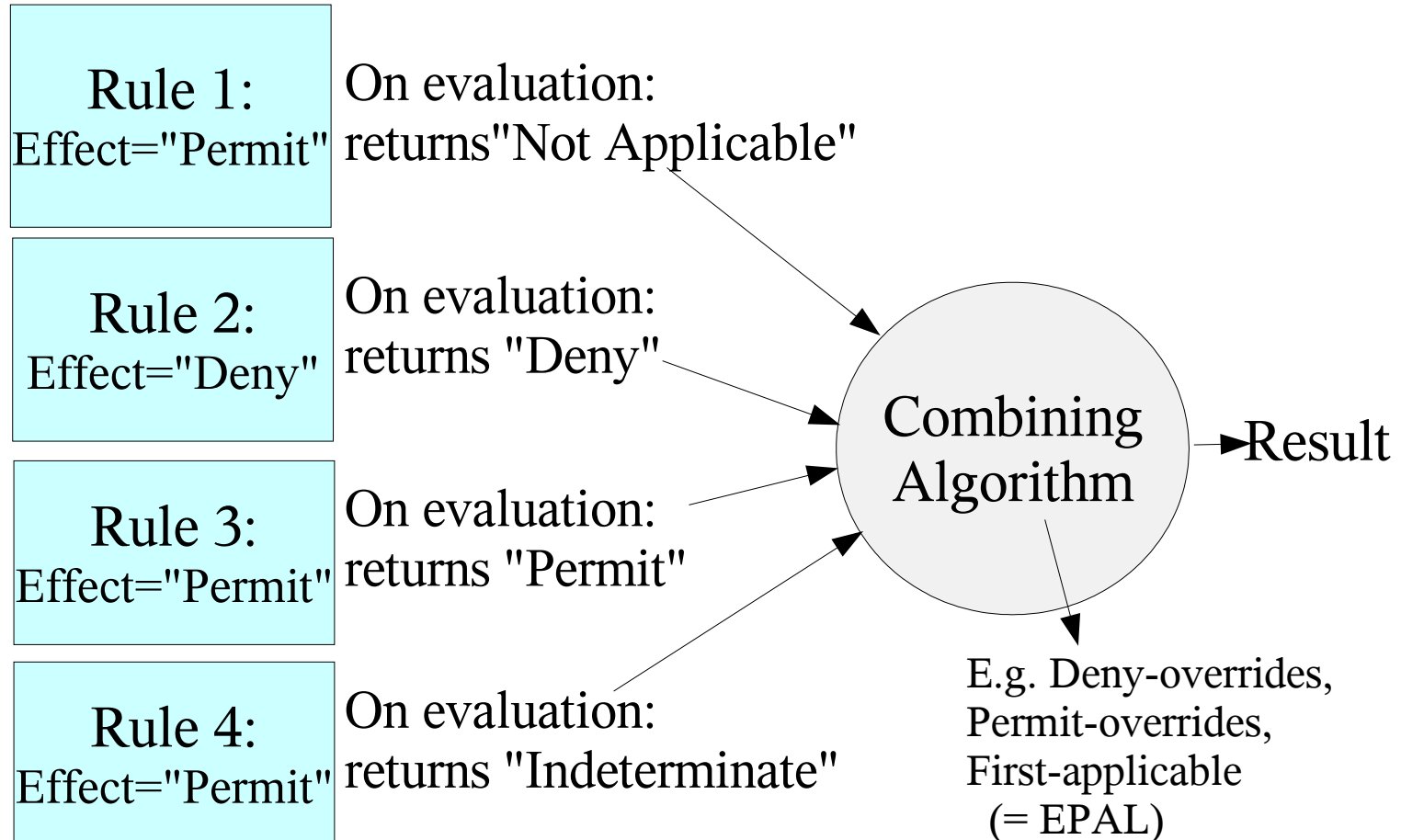
Outline

- Overview
 - XACML
 - EPAL
- Choice of rules to evaluate
- Policy vocabulary
- Hierarchical Request objects
 - XML document resources
- Subjects
- Conclusion

Rules to evaluate:EPAL

- Fixed algorithm:
 - Process rules in document order
 - Return result from first applicable rule whose conditions are satisfied

Rules to evaluate: XACML



Rule choice impact

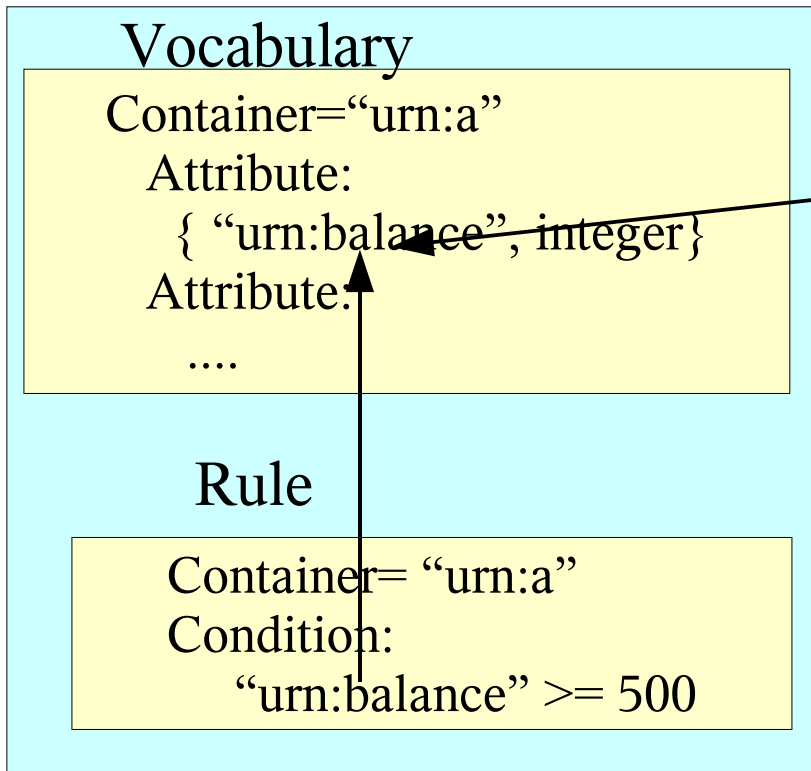
- **First applicable rule**
 - Not scalable – requires global rule preference agreement
 - Sacrifices closure under combination [Barth]
- **Deny unless all applicable rules permit**
 - Forces evaluation of all rules unless “deny” returned earlier: inefficient if “deny” is rare
 - “Deny” rules insecure in open environments
- **Permit if at least one rule permits**
 - Incompatible with “deny” rules
 - Forces evaluation of all rules unless “permit” returned earlier
 - Not scalable – coordination required to ensure deny
- **XACML gives choice; EPAL does not**

Outline

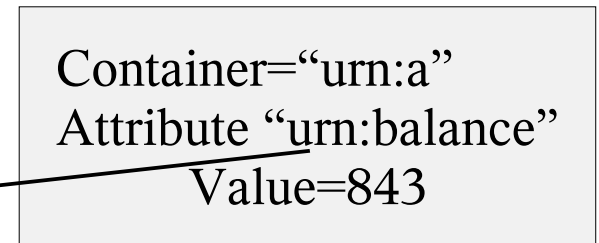
- Overview
 - XACML
 - EPAL
- Choice of rules to evaluate
- Policy vocabulary
- Hierarchical Request objects
 - XML document resources
- Subjects
- Conclusion

EPAL Policy Vocabulary

Policy

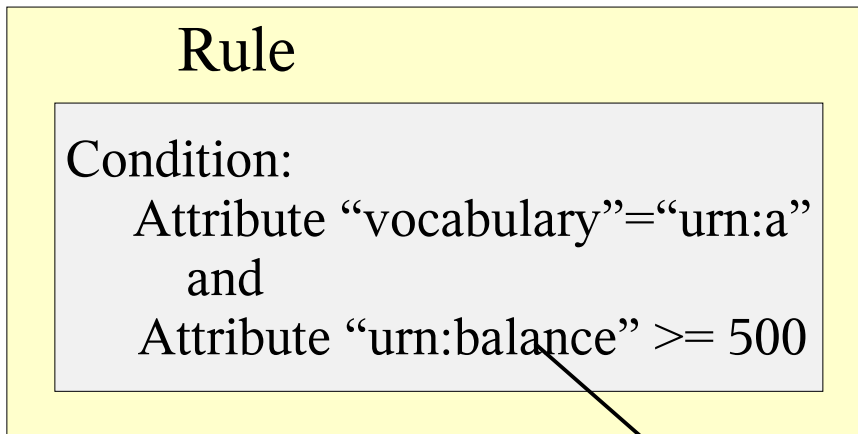


Request Context

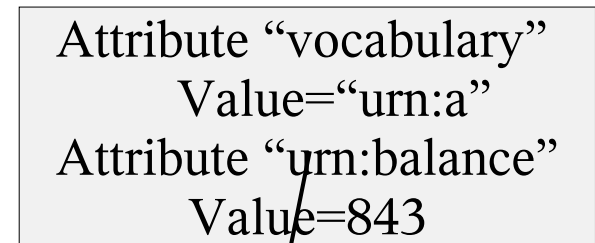


XACML Policy Vocabulary

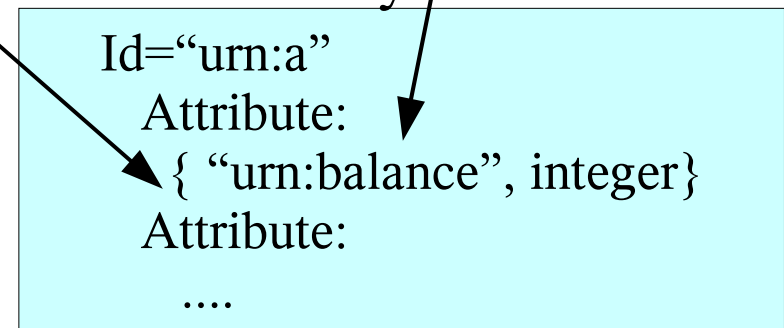
Policy



Request Context



Vocabulary



Vocabulary impact

- **EPAL: Specified in policy**
 - Policy evaluation can enforce agreement
 - Attribute references in Conditions do not need to include DataType, so policies shorter
 - Containers require Requester to know Containers needed for applicable Rules
- **XACML: Specified externally**
 - Vocabulary specifiers are not usually the ones who write policies, so more flexible
 - Request may supply only a subset of Attributes; whether sufficient determined during evaluation
 - XACML references can save re-specifying Attribute details

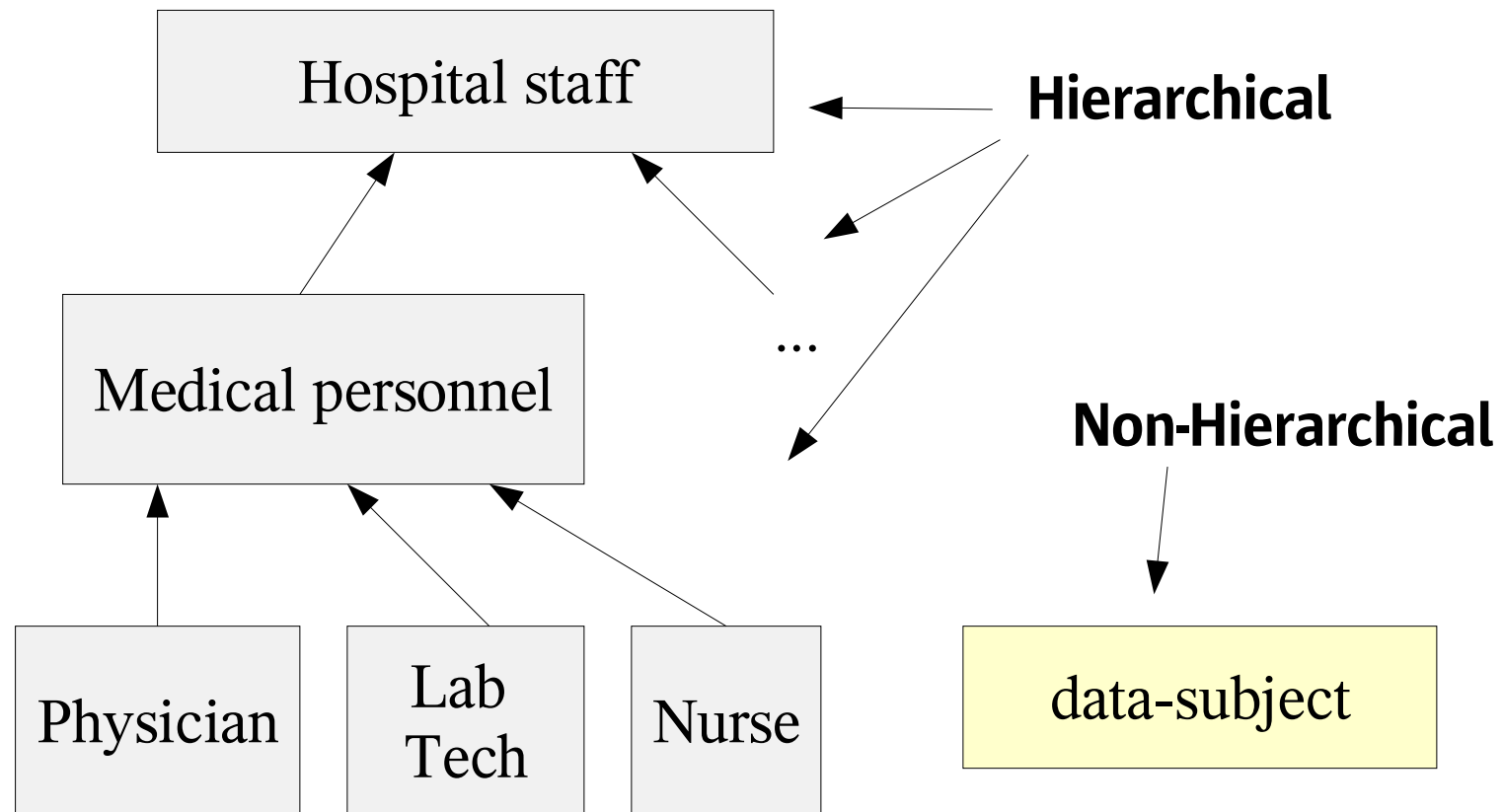
Outline

- Overview
 - XACML
 - EPAL
- Choice of rules to evaluate
- Policy vocabulary
- Hierarchical Request objects
 - XML document resources
- Subjects
- Conclusion

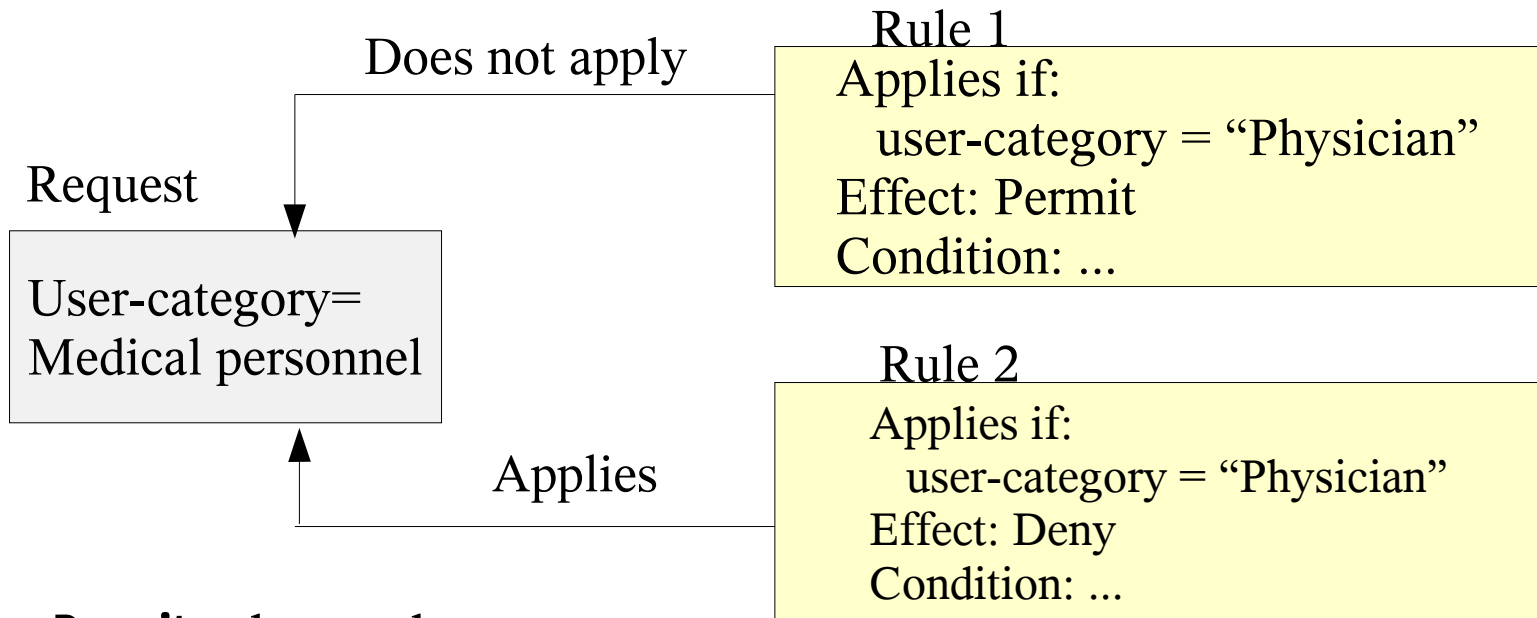
EPAL “categories”

- Two types of Attributes
 - 1) User-category, data-category, purpose-category
 - 2) Container attributes
- Categories
 - Used only for applicability of a Rule
 - At least one Attribute from each category type must be supplied in each Request
- Category Attributes may be hierarchical
- Semantic hierarchies, not physical

EPAL hierarchical vocabulary example: user-categories



EPAL rules and hierarchies



Permit rules apply to:

specified category and its descendants

Deny rules apply to:

specified category, its descendants, and its ancestors

XACML hierarchies

- Hierarchical “roles” (profiled for Subjects)
 - Inherit up: more senior/parent roles are more powerful
 - Senior roles also inherit “deny” permissions of juniors
 - Most senior role must be included in Request
- Hierarchical resources (profiled for Resources)
 - “resource-ancestor” or “scope”=“EntireHierarchy”, “Children”...
 - Constraints can be applied to those Attributes
- Semantic hierarchy possibilities
 - “Category” Attributes: “user-category”, “resource-category”, ... + “...-category-ancestor”, “...-category-descendant”
 - XML categories: XPath expressions to select ancestors or descendants, “xpath-node-match” operator (exists)

XACML: XML doc example

Resource

```
<MedicalRecord>  
  <PatientName>...  
  <PrimaryPhysician>...  
  <History>...  
  <Diagnosis>...  
  <AccountStatus>...  
</MedicalRecord>
```

Policy

Permit access if:

```
  Subject-id=  
/MedicalRecord/PatientName/text()  
  OR  
  { Subject-id=  
/MedicalRecord/PrimaryPhysician/text()  
  AND  
  Resource-id≠  
/MedicalRecord/AccountStatus/* }
```

Hierarchy impact: EPAL

•EPAL

- Models typical privacy policy abstraction levels well
- Hierarchies defined in policy itself
 - But policy writer often not hierarchy definer/owner
- Only user-category, data-category, purpose-category can be hierarchical
- Hierarchies used only for applicability of rules
- Semantic hierarchies, not physical hierarchies
 - Does not handle hierarchical documents (e.g. XML docs)
- Does not handle hierarchical roles (where parents inherit privileges of children)
- Each combination of categories processed separately
 - Combinatorial explosion

Hierarchy impact: XACML

•XACML

- Does not handle semantic hierarchies (where children inherit semantics and privileges of parents)
- Open question: can “category” Attributes and XACML Context Handler handle semantic hierarchies?
- Handles access to subtrees of hierarchical documents
 - One response for entire subtree (if ALL permit), OR
 - Response per node in subtree
- All role, other attributes evaluated in a single pass
- Each node in hierarchical resource document evaluated separately

Outline

- Overview
 - XACML
 - EPAL
- Choice of rules to evaluate
- Policy vocabulary
- Hierarchical Request objects
 - XML document resources
- Subjects
- Conclusion

Subjects: EPAL

- One subject entity per access Request
- Entity may belong to multiple user-categories
- Each user-category evaluated as if submitted separately
 - Can't say “must be in Category A and Category B”
 - If any user-category gets “permit”, return “permit”
 - Secondly, if any user-category gets “deny”, return “deny”

Subjects: XACML

- One or more entities per Request
 - Id of user identified with session
 - Id of application initiating actual Request
 - Id of platform on which application is executing
 - Id of intended receiver of information
 -
- Any subject may have multiple roles
- All entities, roles evaluated at once

Subject handling: impact

•EPAL

- Only one subject entity per Request
- Only “permit-overrides” semantics supported:
 - E.g. Must be in Category A OR in Category B
- Separate evaluation pass for each category

•XACML

- Multiple subject entities all involved with the Request
- Flexible semantics supported:
 - E.g. Must be in Role A AND/OR Role B
- One evaluation pass for all entities, all roles

Outline

- Overview
 - XACML
 - EPAL
- Choice of rules to evaluate
- Policy vocabulary
- Hierarchical Request objects
 - XML document resources
- Subjects
- Conclusion

Conclusion

- XACML is an OASIS Standard
 - EPAL is IBM proprietary specification
- XACML more flexible
 - EPAL is a subset of XACML functionality
 - XACML superset is important for privacy policies, scalability
- Exception: Semantic hierarchies
 - XACML does not support semantic hierarchies; EPAL does
 - Useful in connecting abstract to concrete policies
 - Is policy itself appropriate place to specify semantic hierarchies?
 - Open question: can XACML Context Handler fill the gap?

References

- A. Anderson, *A Comparison of EPAL and XACML*, <http://research.sun.com/projects/xacml/CompareEPALandXACML.html>
- T. Moses, ed., *eXtensible Access Control Markup Language (XACML) v2.0*, OASIS Standard, February 2005, <http://www.oasis-open.org/committees/xacml>
- T. Moses, ed., *Privacy profile of XACML v2.0*, OASIS Standard, February 2005, <http://www.oasis-open.org/committees/xacml>
- C. Powers, et al., eds., *Enterprise Privacy Authorization Language (EPAL 1.2)*, 10 November 2003, <http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/>
- [Barth], A. Barth, J. C. Mitchell, J. Rosenstein, *Conflict and combination in privacy policy languages*, ACM Workshop on Privacy in the Electronic Society, 17 June 2004.



Sun, Sun Microsystems, the Sun logo, and Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and in other countries.

Copyright 2005 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.