

Flexible access control policy languages

Presented by Yuhui Zhong

1

Content

- A unified framework for enforcing multiple access control policies (14 pages)
- XML document security based on provisional authorization (10 pages)
- XACML (22 pages)
- Project (1 page)

2



Flexible support for multiple access control policies (Jajodia et.al ACM Trans. Data. Syst. 26(2))

- Problem statement:
 - Each access control system is bound to a specific policy
 - Single policy cannot capture all the protection requirements
 - Design an unified framework to enforce multiple access control policies within a single system
- Multiple access control policies:
 - Positive vs. negative authorization
 - Conflict resolution
 - Propagation along hierarchy
 - Decision policies
 - Generic constraints

3



Positive vs. negative authorization

- Positive authorization
 - Specify accesses to be allowed
 - Example: ACM, RBAC
 - Represented as (o, s, +a)
- Negative authorization
 - State accesses to be denied
 - Represented as (o, s, -a)

4

Propagation policies

- Hierarchical structures in systems

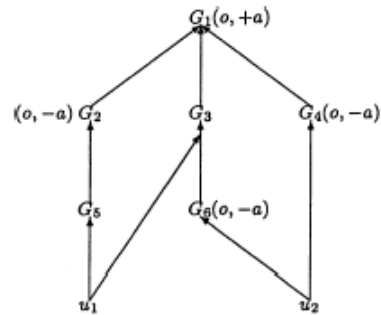
- Object hierarchy

- Examples: file system, XML document

- User-group hierarchy

- Role hierarchy

- An authorization attached to a node can be propagated to its descendants.



5

Propagation policies (cont')

- Propagation policies define how authorizations flow along a hierarchy

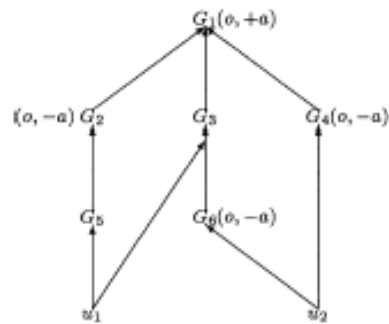
- Examples of propagation policies:

- No propagation
 - No overriding
 - Most specific propagation
 - Path overriding

6

No propagation

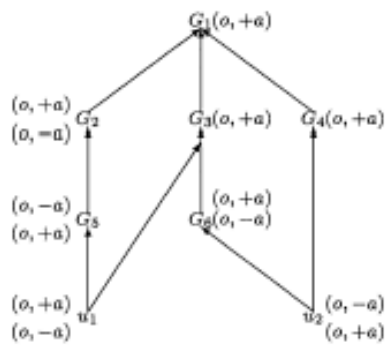
- Authorizations on a node are not propagated to its subnodes



7

No overriding

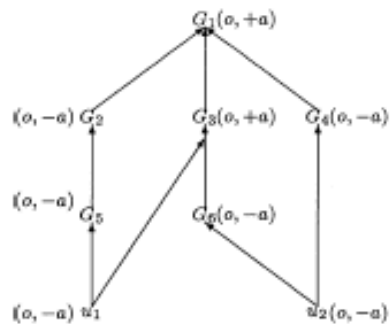
- All the authorizations of a node are propagated to its subnodes
 - Presence of contradicting authorizations



8

Most specific overrides

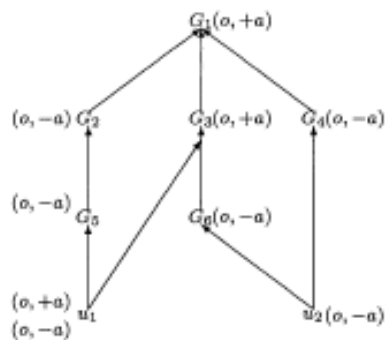
- Authorizations of a node are propagated to its subnodes if not overridden
 - No contradicting authorizations



9

Path overrides

- Authorization attached to a node n overrides a contradicting one from its supernodes on the paths passing from n



10

Conflict resolution policies

- What is conflict?
 - Contradicting authorizations attach to one node
- The reasons conflict occurs?
 - Existence of positive and negative authorization
- Conflict resolution policies
 - No conflicts
 - This policy assumes that no conflicts occur. The presence of conflicts is considered an error.

11

Conflict resolution policies (cont')

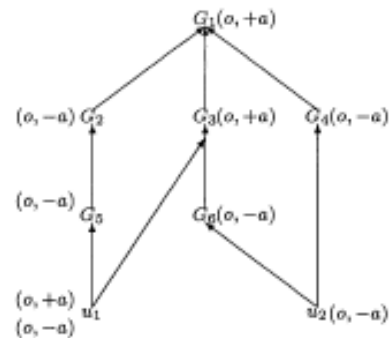
- Denials take precedence
 - Negative authorizations are always adopted when a conflict occurs.
- Permissions take precedence
 - Positive authorizations are always adopted when a conflict occurs
- Nothing takes precedence
 - Neither authorize nor deny an access when there is a conflict.
 - Defer decision to the decision policies

12

An example

- Authorization for u_1 after resolving conflicts

- No conflicts: error
- Denials take precedence: $(o, -a)$
- Permissions take precedence: $(o, +a)$
- Nothing takes precedence: no authorization hold



13

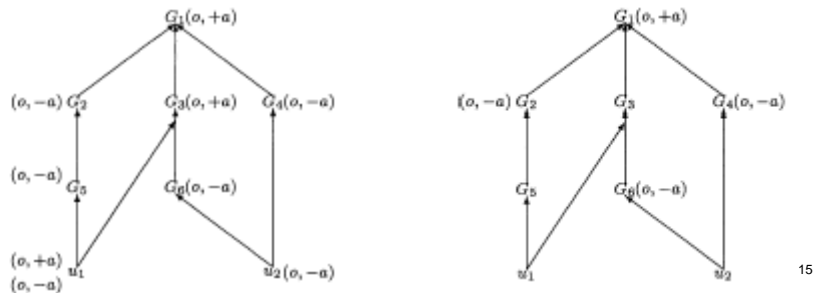
Decision policies

- Apply to the authorizations produced by the propagation and conflict resolution policies
- Force a decision in two cases:
 - No authorization present
 - Unresolved conflicts
- Open policy
 - Deny access if there exists a negative authorization.
 - Otherwise allow it .
- Closed policy
 - Allow an access if a positive authorization exists.
 - Otherwise deny it.

14

An example

- Open policy
 - Left figure: u_1 is denied
 - Right figure: u_1 is allowed
- Close policy
 - Left figure: u_1 is allowed
 - Right figure: u_1 is denied



15

Overview of proposed approach

- A flexible authorization framework
- The framework is based on a authorization specification language (ASL)
- SSO specifies positive or negative authorizations, propagation policies, conflict resolution policies, decision policies, and other integrity rules by using ASL
- These rules consist of a locally stratified logic program
- The unique stable model of the rules can be computed in quadratic time

16

XML document security based on provisional authorization (Kudo and Hada ccs'00)

- In traditional authorization model access request is either granted or denied
- Provisional authorization
 - Access request will be authorized if certain security actions are taken
- Provisional authorization model
 - Authorization policy
 - Authorization request
 - Authorization Decision

17

Authorization policy

- 7- tuple <obj, sub, act, pms, prv, cxt, fml>
 - obj: Xpath expression points to target object
 - sub: user ID of request initiator
 - act: access mode permitted on the object
 - {read, write, create, delete}
 - pms: {grant, deny}
 - prv:
 - {log, verify, encrypt, transform, write, create, delete}
 - cxt: context information such as time, location, and/or arguments for the specific action

18



Authorization policy (cont')

- *fml*: logical conjunction of equalities and/or inequalities
- Semantics of an authorization policy
 - *pms* = 'grant' & *fml* holds: subject can perform the action on the object under the defined context, provided that all of the provisional actions are executed.
 - *pms* = 'deny' & *fml* holds : subject cannot perform the action on the object under the defined context, however all of the provisional actions must still be executed.

19



Authorization request & decision

- Request: <obj, sbj, act, cxt>
- Decision: <obj, sbj, act, pms, prv, cxt>
 - *pms* = 'grant' & prv is not empty: subject is permitted to perform the action on the object under the context, provided all of the provisional actions are executed.
 - *pms* = 'deny' & prv is not empty: request is denied but all of the provisional actions must still be executed.

20



An example

- Authorization policy:
 - <contractor, RegisteredClient, read, deny,log, today is holiday, _>
- Authorization request:
 - <contractor, Registered Client, read, _> submitted in holiday
- Authorization decision:
 - <contractor, Registered Client, read, deny, log, _>

21



XACL

- An access control language based on the provisional authorization model
- <policy> : top-level element
- <xacl>: specifies objects and rules.
- <rule>: <acl> elements
- <acl>: subjects, actions and an optional condition
 - <!ELEMENT policy (property?, xacl)*>
 - <!ELEMENT xacl (object+, rule+)>
 - <!ELEMENT rule (acl)+>
 - <!ELEMENT acl (subject*, privilege+, condition?)>

22



Subject, object, and condition

- Subject: uid, role-set and goroup set

```
<!ELEMENT subject (uid?,role*,group*)>
<!ELEMENT uid (#PCDATA)>
<!ELEMENT role (#PCDATA)>
<!ELEMENT group (#PCDATA)>
```
- Object: element or a set of elments identified by a Xpath expression

```
<!ELEMENT object EMPTY>
<!ATTLIST object href CDATA #REQUIRED>
```
- Conditions

```
<!ELEMENT condition ANY>
```

23



Actions

- “permission” attribute: grant or denial
- “name” attribute: action name
- Provisional action: name and timing
 - Timing: before or after the specified action is executed

```
<!ELEMENT action (parameter*, provision*)>
<!ATTLIST action name (read|write|create|delete) #REQUIRED
                permission (grant|deny) #REQUIRED>
<!ELEMENT provisional_action (parameter*)>
<!ATTLIST provisional_action name CDATA #REQUIRED
                timing (before|after) "after">
```

24



<property> element

- <!ELEMENT property (propagation?, conflict-resolution?, default?)>
- Three types of propagation policy
 - <!ELEMENT propagation EMPTY>
 - <!ATTLIST propagation read (no|up|down) "down"
write (no|up|down) "down"
create (no|up|down) "down"
delete (no|up|down) "up">

25



<property> element

- Three types of conflict resolution policy
 - <!ELEMENT conflict-resolution EMPTY>
 - <!ATTLIST conflict-resolution read (dtp|ptp|ntp) "dtp"
write (dtp|ptp|ntp) "dtp"
create (dtp|ptp|ntp) "dtp"
delete (dtp|ptp|ntp) "dtp">
 - ntp: refer to the default policy when conflict occurs
- Default policies for four actions
 - <!ELEMENT default EMPTY>
 - <!ATTLIST default read (grant|denial) "denial"
write (grant|denial) "denial"
create (grant|denial) "denial"
delete (grant|denial) "denial">

26



XACML

- Data flow model (3 pages)
- Language (11 pages)
 - Policy schema
 - Context schema
- Functional requirements (6 pages)
- Other issues (1 page)

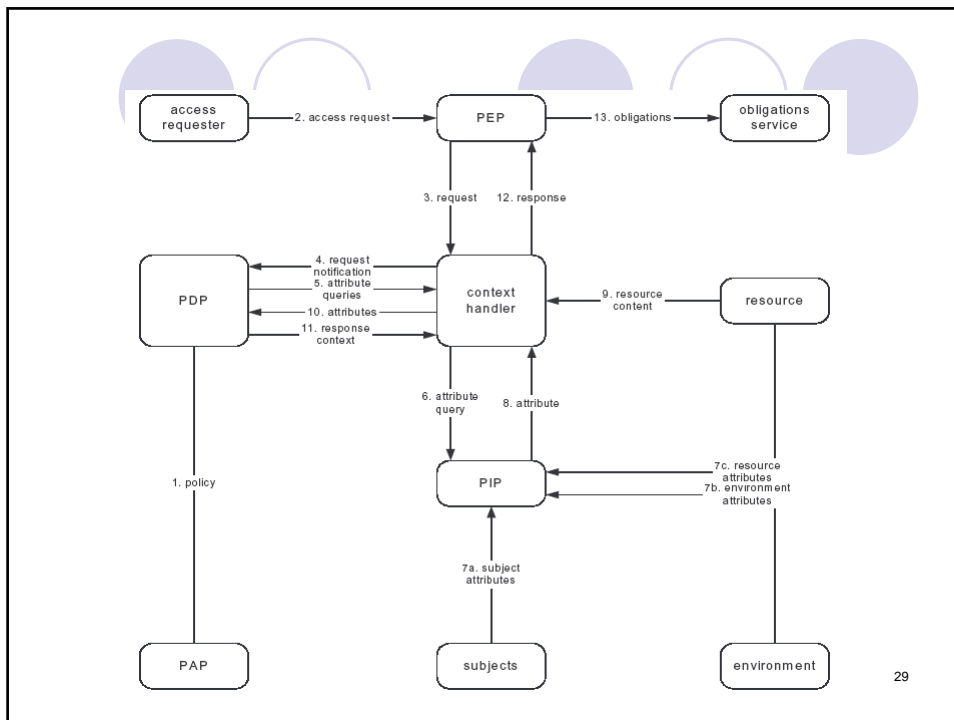
27



XACML Data flow model

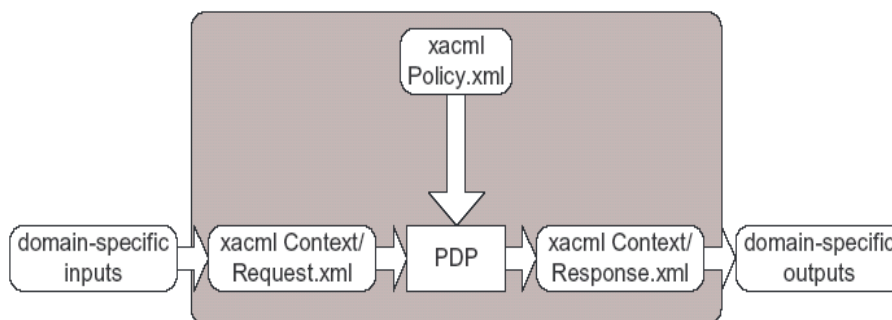
- Policy Administration Point (PAP)
 - Policy creation
- Policy Enforcement Point (PEP)
 - Decision enforcement
- Policy Information Point (PIP)
 - Attribute or data provider
- Policy Decision Point (PDP)
 - Policy evaluation & decision making

28



Context

- Canonical representation of request and responds



Language – Policy syntax

- PolicySet: top-level element
 - A target element
 - Defined by creator or computed from the <Target> elements of <policy> elements
 - A policy-combining algorithm identifier
 - Deny-overrides (ordered or unordered)
 - Permit-overrides (ordered or unordered)
 - First-applicable
 - Only-one-applicable
 - A set of policies
 - Obligations
 - Provisional actions
 - No standard definitions for these actions. Bilateral agreement between PEP and PDP

31

Language (cont')

- Policy: smallest entity that shall be presented to the PDP for evaluation
 - A target element
 - A rule-combining algorithm identifier
 - Deny-overrides (ordered or unordered)
 - Permit-overrides (ordered or unordered)
 - First-applicable
 - A set of rules
 - Obligations

32

Language (cont')

- Rule: most elementary unit of policy
 - A target element
 - The set of resource, subjects and actions to which the rule is intended to apply
 - If absent, inherent target from policy
 - An effect attribute
 - Permit or deny
 - A condition element
 - Refine the applicability of the rule

33

Target and subjects

- Target:
 - A conjunctive sequence of <Subjects>, <Resources> and <Actions> elements
- Subjects:
 - <AnySubject> Or a disjunctive sequence of <Subject>
- Subject:
 - A conjunctive sequence of <SubjectMatch>
- SubjectMatch:
 - <AttributeValue>: an embedded attribute value
 - <AttributeSelector> or <SubjectAttributeDesignator>: Attribute values in a <Subject> element of request context
 - MatchId: specify a matching function

34

Resources and actions

- Resources:
 - <AnyResource> Or a disjunctive sequence of <Resource>
- Resource :
 - A conjunctive sequence of < ResourceMatch>
- ResourceMatch:
 - <AttributeValue>
 - <AttributeSelector> or <ResourceAttributeDesignator>
 - MatchId
- Actions:
 - <AnyResource> Or a disjunctive sequence of <Action>
- Action:
 - A conjunctive sequence of < ActionMatch>
- ActionMatch:
 - <AttributeValue>
 - <AttributeSelector> or < ActionAttributeDesignator>
 - MatchId

35

Matching functions

- {=, >, <, ≥, ≤, match} on primitive types
 - Primitive types: {string, boolean, integer, double, time, date, dateTime, anyURI, hexBinary, base64Binary, dayTimeDuration, yearMonthDuration, x500Name, rfc822Name}
- Evaluation semantics:
 - Operational error: Indeterminate
 - <AttributeDesignator> or <AttributeSelector> element evaluate to an empty bag: false
 - Apply Matching function between the embedded attribute value and each returned elements
 - One application evaluates to True: True
 - One application results in Indeterminate: Indeterminate
 - All applications evaluate to False: False

36

Condition

- a boolean function over subject, resource, action and environment attributes or functions of attributes.
- ApplyType
 - FunctionId
 - Function:
 - A function applied to the elements of a bag
 - Apply:
 - A nested function-call argument
 - Argument:
 - literal value, subject attribute, resource attribute, action attribute, environment attribute

37

XACML standard functions

- Equality predicates
 - rfc822Name, x500Name, anyURI
- Logical functions
 - {or, and, n-of, not}
- Arithmetic and non-numeric comparison functions
 - {=, >, <, ≥, ≤}
- Special math functions
 - {regexp-string-match, x500Name-match, rfc822Name-match}
- XPath-based functions
 - {xpaht-node-count, xpath-node-equal, xpath-node-match}

38

XACML standard functions (cont')

- Arithmetic functions
 - {+, -, *, /, %, abs, round, floor}
- Date and time arithmetic functions
 - {+, -}
- String conversion functions
- Numeric data-type conversion functions
- Bag functions
 - {one-and-only, bag-size, is-in, bag}
- Set functions
 - { \cap , \cup , \subset , =, at-least-one-member-of}
- Higher-order bag functions

39

Obligations

- A set of <Obligation> elements
- <Obligation>
 - ObligationId
 - FulfillOn
 - The effect for which this obligation applies
 - AttributeAssignment
 - Obligation arguments assignment.

40

Language model - Context

- Request: top-level element in context schema
 - One or more <Subject> elements
 - A <Resource> element
 - An <Action> element
 - An <Environment> element
- Respond: top-level element in context
 - One or more <Result> element
- Result: An authorization decision result
 - <Decision>: {Permit, Deny, Indeterminate, NotApplicable}
 - <Status>: whether errors occurred
 - <Obligations>

41

Functional requirements

- PEP
 - Access is allowed only if a valid XACML response of "Permit" is returned by PDP.
 - A response of "Permit" is valid only if the PEP understands all of the obligations.
- Condition evaluation
 - True: <Condition> element is absent or if it evaluates to "True" for the attribute values supplied in the request context.
 - False: <Condition> element evaluates to "False" for the attribute values supplied in the request context
 - Indeterminate : if any attribute value referenced in the condition cannot be obtained.

42

Target evaluation

- Match: subject, resource and action specified in the target all match values in the request context
- No-match: one or more of the subject, resource and action specified in the target do not match values in the request context.
- If referenced attribute value is not available
 - MustBePresent is true: Indeterminate
 - MustBePresent is false: No-match

43

Rule evaluation

Target	Condition	Rule value
Match	True	Effect
Match	False	NotApplicable
Match	Indeterminate	Indeterminate
No-match	Don't care	NotApplicable
Indeterminate	Don't care	Indeterminate

44

Policy evaluation


Target	Rule values	Policy value
Match	At least one rule value is its Effect	Specified by the rule-combining algorithm
Match	All rule values are NotApplicable	NotApplicable
Match	At least one rule value is Indeterminate	Specified by the rule-combining algorithm
No-match	Don't care	NotApplicable
Indeterminate	Don't care	Indeterminate

45

Policy set evaluation

Target	Policy values	Policy set value
Match	At least one policy value is its Decision	Specified by the policy-combining algorithm
Match	All policy values are NotApplicable	NotApplicable
Match	At least one rule value is Indeterminate	Specified by the policy-combining algorithm
No-match	Don't care	NotApplicable
Indeterminate	Don't care	Indeterminate

46



Obligations

- An obligation SHALL be passed up to the next level of evaluation only if the effect of the policy or policy set being evaluated matches the value of the xacml:FulfillOn attribute of the obligation.
- PEP receives a valid XACML response of "Permit" with obligations: fulfill all of those obligations.
- PEP receives an XACML response of "Deny" with obligations: fulfill all of the obligations that it understands.

47



Misc

- Implementations
 - Jiffy Software
 - Sun
- Related Proposals and drafts
 - XACML RBAC profile
 - XACML profile for Web-services
 - XACML XML DSig Profile
 - LDAP profile for distribution of XACML policies

48



A logic foundation for XACML

- Transform a XACML policy set into Constraint datalog program.
- Identify a subset of XACML policy that can be translated into Constraint Datalog
- Translate a policy set written by using the above subset of XACML into Constraint Datalog
 - Negation
 - Propagation
 - Conflict management
 - Condition: The first part of XACML functions
 - Data type: anyURI, x500Name, rfc822Name
- Security analysis problem

49