XACML: Formalization and Analysis

Analyzing Web Access Control Policies

Vladimir Kolovski
James Hendler, Bijan Parsia

-Ankesh
Outline

- XACML Overview
- XACML Formalization
- XACML Policy Analysis Services
Policy - declarative set of rules that guide the behavior of entities within a system (editorial JWS 2009)

XACML OVERVIEW
Example

- **PS1 First-App**
  - **P1 Permit-Override**
    - **R1** (Manager) (write) OR (read) (Report) Permit
    - **R2** (Developer) (read) (Report) Permit
    - **R3** (Any) (Any) Deny
  - **PS2 Permit-Override**
    - **P2 Permit-Override**
      - **R4** (Developer) (write) (Report) Permit
Attribute – value in **Predicate**:
- **Role** (of subject): developer
- **Action-type**: read
- **Resource-type**: Report

**Target**: Set of decision requests, identified by definitions for resource, subject and action, that a rule, policy or policy set is intended to evaluate
Combining Algorithms

Additional Features- Role hierarchies (eg. Role A is senior to role B)
- Separation of role constraint.
Defeasible Description Logic
Capture the behavior of XACML language

FORMALIZING XACML
In Past...

- LP systems common choice for formalization

- Formalization of RBAC based on ALCQ DL (‘Representation and reasoning on rbac: A description logic approach’ - ICTAC 2005)
  - XACML is more expressive
  - SHOIN

- Formalization of RBAC using multi-modal logic (‘Reasoning about security: A logic and a decision method for role-based access control’ - ECSQARU-FAPR 1997)
  - Decision method based on analytic tableaux
Overview

• To capture behavior of XACML

• Basic unit that yields an access decision – Rule
• Formalize prerequisite and head of rules
• How access decision is propagated upwards to the root PolicySet
• Combining algorithms need formalism that supports preferences

• Target → DL concept expr.

• Rule/ Policy preferences → Defeasible rules
Defeasible Description Logics

- *(wiki) Defeasible reasoning* is the study of forms of reasoning that, while convincing, are not as formal and rigorous as deductive reasoning.
  - Rule represented as $\text{Pre}(r) \rightarrow \text{Con}(r)$
  - Rule Set = Strict-rules $\cup$ Defeasible-rules
  - Defeasible rules are *rules that can be defeated by contrary evidence!*
    - Eg. “Lecturers are faculty member”
  - Superiority relation is represented as $r > r'$

- $\text{DDL}^- \subseteq \text{DDL}$ – same computational complexity as underlying DLs.
- $\text{DDL}^-$ - no DL KB predicate in the head
\[ P_1 : \text{Permit-}P_1 \leftrightarrow \text{Permit-}PS1, \]
\[ P_2 : \text{Deny-}P_1 \leftrightarrow \text{Deny-}PS1 \]
\[ P_3 : \text{Permit-}PS2 \leftrightarrow \text{Permit-}PS1, \]
\[ P_4 : \text{Deny-}PS2 \leftrightarrow \text{Deny-}PS1, \]

\[ R_1 : \exists \text{role-type.Manager} \cap \exists \text{res-type.Report} \cap \]
\[ (\exists \text{action - type.read} \cup \exists \text{action-type.write}) \leftrightarrow \text{Permit-}P_1 \]
\[ R_2 : \exists \text{role-type.Developer} \cap \exists \text{action-type.read} \]
\[ \exists \text{res-type.Report} \leftrightarrow \text{Permit-}P_1 \]
\[ R_3 : \exists \text{role.Manager} \uplus \exists \text{role.Developer} \uplus \exists \text{action.Write} \uplus \]
\[ \exists \text{action.read} \uplus \exists \text{resource.Report} \leftrightarrow \text{Deny-}P_1 \]
\[
\delta = \{ P_1 > P_4, P_2 > P_3, R_1 > R_3, R_2 > R_3 \}
\]
• XACML requests are mapped to DL-term in same manner as rules (as a list of attribute value pairs)

• Matching request $r$ with target $T \equiv$ checking whether $r$ is an instance of $\text{DL-Term}(T)$
Map($\alpha$)

**Definition 4.** If $a \notin L$, then $map(a) = a$. When $a \in L$, there are two cases - it is a Permit effect-literal or a Deny effect-literal. For a Permit effect-literal,

$$map(a) = \bigcup (map(C) \cap \neg(\bigcup map(J)))$$

where

- $\{C \mapsto \text{Permit} - P\} \in D$
- $\{J \mapsto \text{Deny} - P\} \in D$
- $J > C$

for some Permit - P, Deny - P.

$map(a)$ is defined analogously for Deny-literals.

- Derivability in DDL$^-$ $\implies$ Concept Satisfiability in DL
  - For an effect-literal $\alpha$ we can create a DL concept expression $A$ ($map(\alpha)$) in $K$ s.t. $\alpha$ is derivable in $D$ iff $A$ is satisfiable in $K$
& now...

• To check if request r is permitted under policy P we would like to check if Permit-P(r) holds, i.e. check whether r is an instance of concept map(Permit-P) !!!!!!!
Policy Analysis
Semantic Descriptions of Policy Domain

BENEFITS FROM MAPPING TO DL
Semantic Descriptions of Policy Domain

- Role hierarchies
  \[ \exists \text{role. LeadDeveloper} \sqsubseteq \exists \text{role. Developer} \]

- Attribute hierarchies
  \[ \text{CIO-of} \sqsubseteq \text{employee-of} \]

- Separation of duty
  \[ \exists \text{role. A} \sqsubseteq \neg \exists \text{role. B} \]

- Cardinality constraints
  \[
  \begin{align*}
  \geq k \text{ role. T} & \sqsubseteq \bot \\
  \geq k \text{ role}^{-} . \text{T} & \sqsubseteq \bot
  \end{align*}
  \]
Analysis Services

• Policy Comparison –
  \[ P_1 \subseteq P_2 \ ? \equiv \ \text{is} \ \text{map}(\text{Permit-}P_1) \cap \neg\text{map}(\text{Permit-}P_2) \ \text{satisfiable} \]

• Policy Verification (verification of policy properties). Eg. Policy P has following property:

  User who is not a Manager and is less than 21 years old is not allowed to perform more than 1 action at a time

\[ \exists \text{role.} \neg \text{Manager} \cap \exists \text{age.} \leq 21 \cap >1\text{action} \rightarrow \text{Deny} \]

Is (Map(Permit-P) \cap A) satisfiable?
More Analysis Services

- Policy Redundancy

1. rule \( r : T \mapsto \text{Permit-P} \)
2. Let \( J = \bigcup \{ \text{map}(j) \mid j > r, j \mapsto C \in R, C \in \{\text{Permit-P}, \text{Deny-P}\} \} \)
3. If \( \text{map}(T) \subseteq J \), then rule is redundant

- For computation of \( J \), we may consider not just \( P \) but also \( Q \) s.t. \( P \mapsto Q \in R \)
Earlier Efforts

• Formalization of RBAC using multi-modal logic; Decision method based on analytic tableaux (‘Reasoning about security: A logic and a decision method for role-based access control’- ECSQARU-FAPR 1997)
  – Provides similar services- Logical consequence, model generation, consistency checking of policies

• LP systems
  – Provided a set of services not offered by rule based policy systems

  – Unclear whether feasible for larger policies
Summary +

Thanks!

• Formalization of an Access Control Policy Language, by mapping to DL (DDL-)

• A(accountability)IR...
  – Acc vs AC
  – Rules vs con(r) not just deny or permit
  – order vs ‘alt’

• A bunch of services for quality checking and performance evaluation of policies
Future Work

• Extend coverage of XACML
  – Only-one-applicable combining algorithm
  – Handle attribute functions

• Currently support for xsd datetime and integer datatypes and common facets (min & max-inclusive)
  – More expressive datetime support to allow periodicity constraints
References

Citations

• Alex X. Liu, Fei Chen, JeeHyun Hwang, Tao Xie, Xengine: a fast and scalable XACML policy evaluation engine, ACM SIGMETRICS Performance Evaluation Review, v.36 n.1, June 2008