A Logic-Based Framework for Distributed Access Control

Vladimir Kolovski

Oracle New England Development Center

1 Oracle Drive, Nashua, NH

Characteristics of Distributed Access Policies

- Attribute-based
  - Identity of users not always known
- Heterogeneous
  - Different protection requirements
  - Rich data-type support, conflict resolution mechanisms
- Distributed
  - References between policies
- Policy Language Proposals
  - Industry: EPAL (IBM), XACML (Sun), SecPal (MSFT)
  - Academia: Cassandra (Becker 2006), RT (Li 2003), FAF (Jajodia 2001), Lithium (Weissman 2003), DL (Li 2001), Rei(n) (Kagal 2003)

eXtensible Access Control Markup Language (XACML)

- Language with a lot of momentum
  - OASIS standard since 2003
  - Supports distributed policies, data-types, conflict resolution
- Industry interest
  - 65 public products and deployments that make substantial use of XACML
- Academic interest
  - 200+ papers citing the XACML Standard

Motivation(1): Lack of a Logic-Based

- XACML lacks an official formal semantics
  - Unclear and ambiguous specification
    - Especially newer features
  - Unknown complexity properties
    - Is access request checking even tractable?
    - Want to know which features cause problems
  - Want to compare and extend XACML
    - Research work in logic-based access control
    - Experiment with adding new features

Motivation(2): XACML Policies Hard to

"When I sat down to support complex policy requirements in a real-world application using a custom database and attribute retrieval system, it was hard....Just understanding the implications of all the policy references and each target on a rule took a lot of effort."

Motivation(2): XACML Policies Hard to

"When I sat down to support complex policy requirements in a real-world application using a custom database and attribute retrieval system, it was hard....Just understanding the implications of all the policy references and each target on a rule took a lot of effort."

-Seth Proctor, one of the designers of XACML

Research Contribution

A logic-based framework that provides a theoretical foundation for XACML and a practical set of static analysis services that cover heterogeneous and distributed XACML policies

Logic-Based Foundation for XACML

Approach: Use Datalog to Formalize XACML

- Datalog is a query and rule language for deductive databases
  - A Datalog program consists of rules and facts
- Desirable computational properties
- Foundation for many access control languages
  - SecPal [Becker2006], FAF [Jajodia2001], Delegation Logic [Li2001], RT [Li2003], PeerTrust [Nejdl2004], etc.

Mapping XACML to Datalog(1)

1. **Generate facts (extensional predicates) from policy structure**

```
<Policy PolicyId="policy1" RuleCombiningAlgId="…rule-combining-algorithm:first-applicable">
  <Target>
    <DisjunctiveMatch>
      <ConjunctiveMatch>
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="…XMLSchema#string">admin</AttributeValue>
          <AttributeDesignator Category="…subject-category:a … le"
         DataType="…XMLSchema#string"/>
        </Match>
      </ConjunctiveMatch>
    </DisjunctiveMatch>
  </Target>
  <Rule RuleId="rule1" Effect="Permit">  <Target/>
   </Rule>
</Policy>
```

Generated predicates:

hasRule(policy1, rule1)

hasTarget(policy1, target-id)

hasEffect(rule1, Permit)

hasMatch(target-id, match-id)

hasMatchFunction(match-id, ‘string-equal’), …

Mapping XACML to Datalog (2)

2. Datalog rules to match access requests against Targets (10 rules)

Example:

matchAD(?AD; ?RQ; ?V) :– hasAttribute(?RQ; ?AT), hasValue(?AT; ?V)
hasAttrID(?AD; ?id), hasAttrID(?AT; ?id)
hasCat(?AT; ?cat), hasCat(?AD; ?cat).

matchM(?M; ?RQ) :– matchAD(?AD; ?RQ; ?V), hasValue(?M; ?VM);
fcn(?V; ?VM) = True.

3. Predicates for access decisions

- PolicySets: *Permit-PS(?X, ?RQ), Deny-PS(?X, ?RQ)*
- Policies: *Permit-P(?X, ?RQ), Deny-P(?X, ?RQ)*
- Rules: *Permit-R(?X, ?RQ), Deny-R(?X, ?RQ)*

Mapping XACML to Datalog (3)

4. **Generate Datalog rules to propagate access decisions**
   - **Propagate from Rules to Policies and PolicySets**
   - **Each combining algorithm provides a different set of propagation rules**
   - **Example of a Permit-overrides propagation:**

```
Deny-P(?P, ?RQ) :-    hasTarget(?P, ?T), matchT(?T, ?RQ)
                           hasRule(?P,?R),  Deny-R(?R, ?RQ)
                           hasComb(?P, Permit-Overrides),
                      :Permit-P(?P, ?RQ).
```

5. **Translate each request RQ to a set of facts and run against Datalog KB**

Mapping Results

- Mapping XACML *Policies* and *Rules* produces a *locally stratified* Datalog program
  - Ordering:

    Match predicates < *Rule* predicates < *Policy* predicates
- Cyclical references between PolicySets break stratifiability restriction
  - Multiple models (or no model) possible, depending on order of evaluation
    - Ambiguous policies!
  - Disallowing cyclical PolicySet references brings XACML down do polynomial complexity

Mapping Implications

- **Compared XACML to well-studied Datalog-based policy frameworks**
  - **Flexible Authorization Framework [Jajodia2001], SecPal [Becker2006] (more)**
- **Can extend XACML with features from other languages without sacrificing complexity**
  - **E.g.,** role hierarchies currently implicit in policy rules
    - Results in incomplete hierarchy support

Example* of incomplete role hierarchy support

- Three roles: Doctor, Nurse, Admin
  - Doctor role is senior to Nurse and Admin
- Two permission sets (for Nurse and Admin)
- Consider a new permission (RegisterNewPatient) is added
  - RegisterNewPatient requires users to activate both Nurse and Admin role
  - XACML will not automatically infer that Doctors are linked to this new permission
- Solution
  - Separate role hierarchy information from policy in XACML
  - Extend semantics by augmenting Datalog mapping with role hierarchy rules

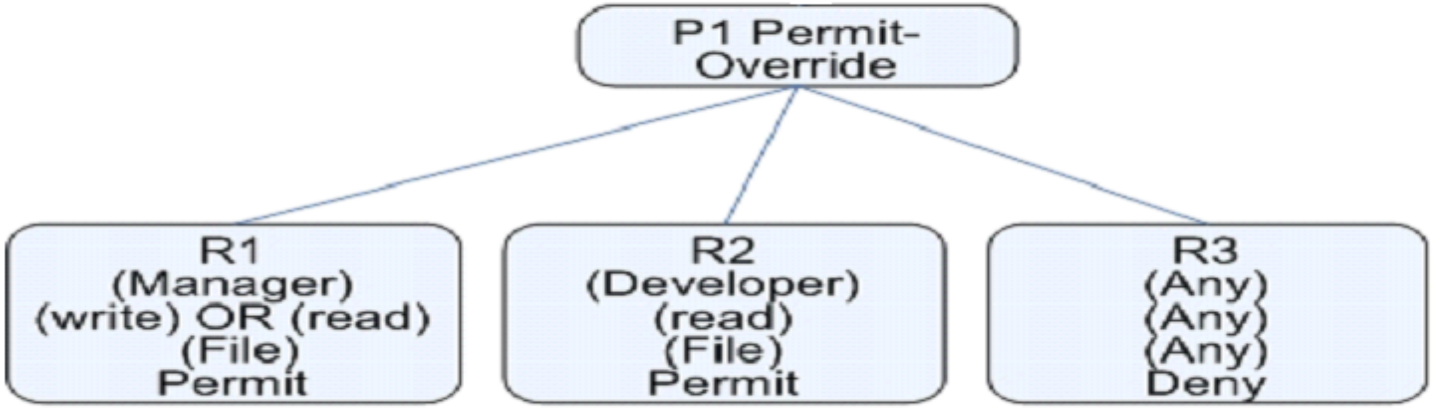
***_D. J. Power et al_. On XACML, role-based access control, and health grids. The 4th UK e-Science AHM, 2005.**

Practical Analysis Services for Policies

Problem: Analysis of XACML Policies

- Interest in providing static analysis services for XACML
- Previous work with limited expressiveness
  - Lacks support for delegation, data-types, policy vocabularies
  - Cannot analyze distributed and heterogeneous policies
- Contribution: developed static analysis framework for expressive XACML policies
  - Provided formal verification, change analysis, reachability analysis, checking for disjoint policies, etc.

Testing vs Formal Verification



- Test case: *Developers* are not allowed to *write* to *File*
- Testing not exhaustive
  - E.g., *Developer* requests to both *write* to and *read* from *File*

Approach: OWL-DL for XACML Analysis

- Web Ontology Language (OWL)
  - Language for representing the semantics of information on the Web
- Developed through the W3C Semantic Web initiative
  - W3C published OWL as a recommendation (Feb 2004)
- Design based on Web architecture
- Comes in three different levels: Lite, DL, Full

Why OWL-DL for XACML Analysis?

- Policy analysis services reduced to DL reasoning tasks
  - Exist off-the-shelf DL reasoners *optimized* for those tasks
    - Pellet, FaCT++, RacerPro, KAON2

- Web-based nature of OWL great fit for XACML
- OWL provides support for rich policy domain modeling and interoperability
  - Already interest in semantic-enabled XACML [Priebe06, Damiani04]

Mapping XACML to OWL-DL: Overview

- Access requests are mapped to OWL individuals
  - XACML attributes -> OWL properties
  - XACML values -> OWL datatype values
- Rules, Policies and PolicySets mapped to OWL classes

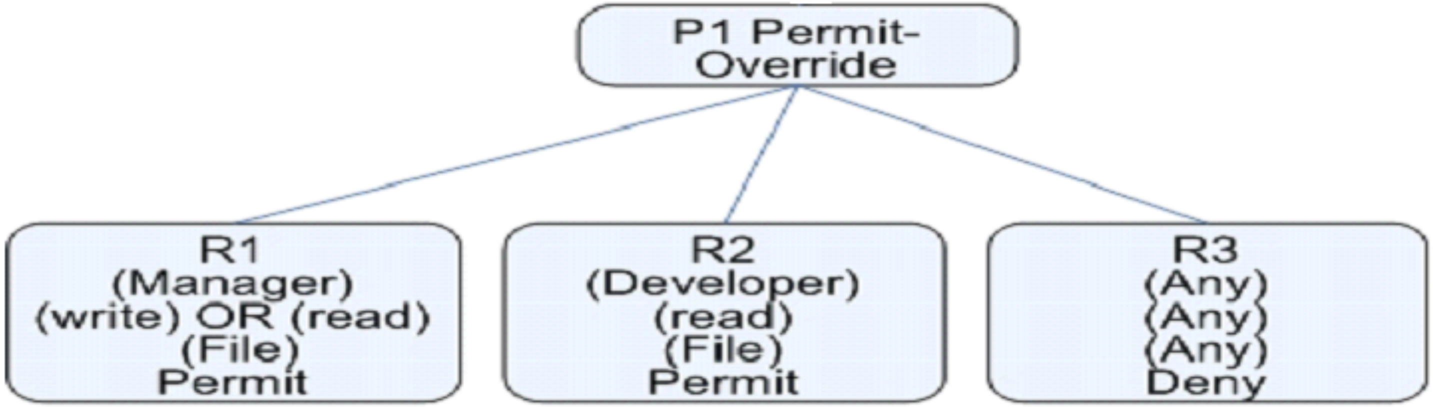
- Generate OWL classes to capture XACML access decisions
  - E.g., for each Rule R: *Permit-R*, *Deny-R* classes
  - Combine concepts: *Permit-R1* u :*Deny-R2*
- Propagate access decisions using subclass and equivalence axioms
  - Depending on combining algorithm

***For details see: Vladimir Kolovski et al. Analyzing Web Access Control Policies. In Proceedings of the 16th International World Wide Web Conference (WWW 2007), 2007.**

Formal Verification

- Used DL concept satisfiability checking for verification
  - DL concept generated based on input policy, test case and expected outcome
- If test fails, extract counter example from model
  - Return access request that causes test failure
- Extract policy *trace*
  - Return a list of policies that fired and produced test failure

Formal Verification Example



- **Test case:**
  - *role=Developer, action=write, resource=File; outcome=NeverPermit*
- **Counterexample:**
  - role=Developer, action=read, action=write, resource=File
- **Policy trace:**
  - R2 (Permit) -> P1 (Permit)

Change Analysis

- Policy diffing
  - Example: Are there any requests where policy P1 returns Deny, and P2 returns Permit?
- Also, **verify** changes
  - Example: Verify that Deny-to-Permit changes do not involve role *Developer*?
- Reduced to satisfiability checking
  - OWL-DL reasoners optimized for this service

Additional Analysis Services

- Reachability Analysis (redundancy checking)
  - Check if a policy is “dominated” by others
  - Can be used to optimize policy engines
- Disjointness
  - Verify that no request applies to both policies
- Explanation for policy errors
  - Leverage OWL-DL debugging support

Analyzing Web Service Policies

Applying Analysis Framework to Web Services

- Web Service Policies
  - Specify constraints and capabilities of web service providers and clients
- WS-Policy is becoming a W3C standard
  - WS-XACML provides a language for WS-Policy assertions
- Policies are mapped to OWL-DL class expressions
  - Analysis services: verification, change analysis, consistency checking

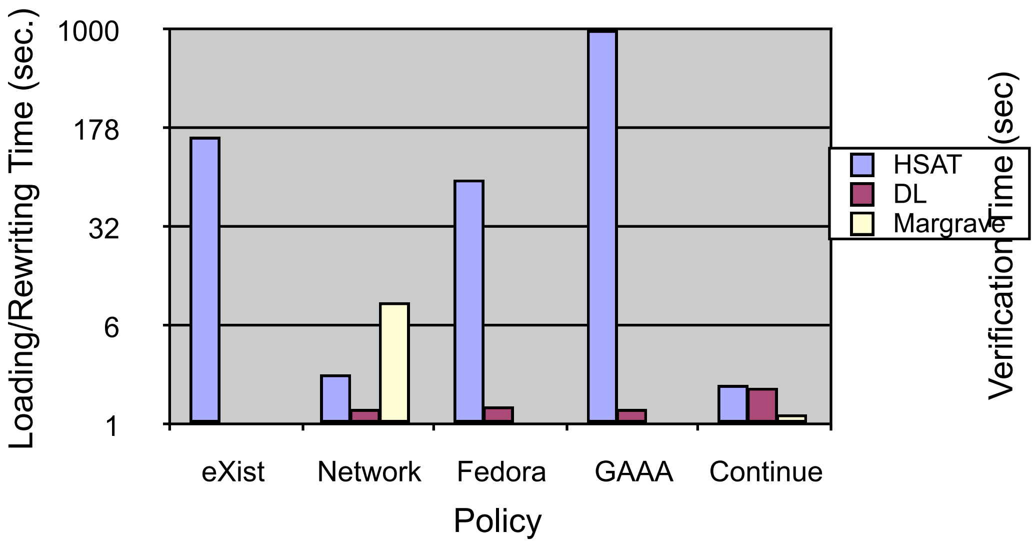
*For details see: Vladimir Kolovski et al. Representing Web Service Policies in OWL DL. *In Proc. of the 4th International Semantic Web Conference (ISWC), 2005.*

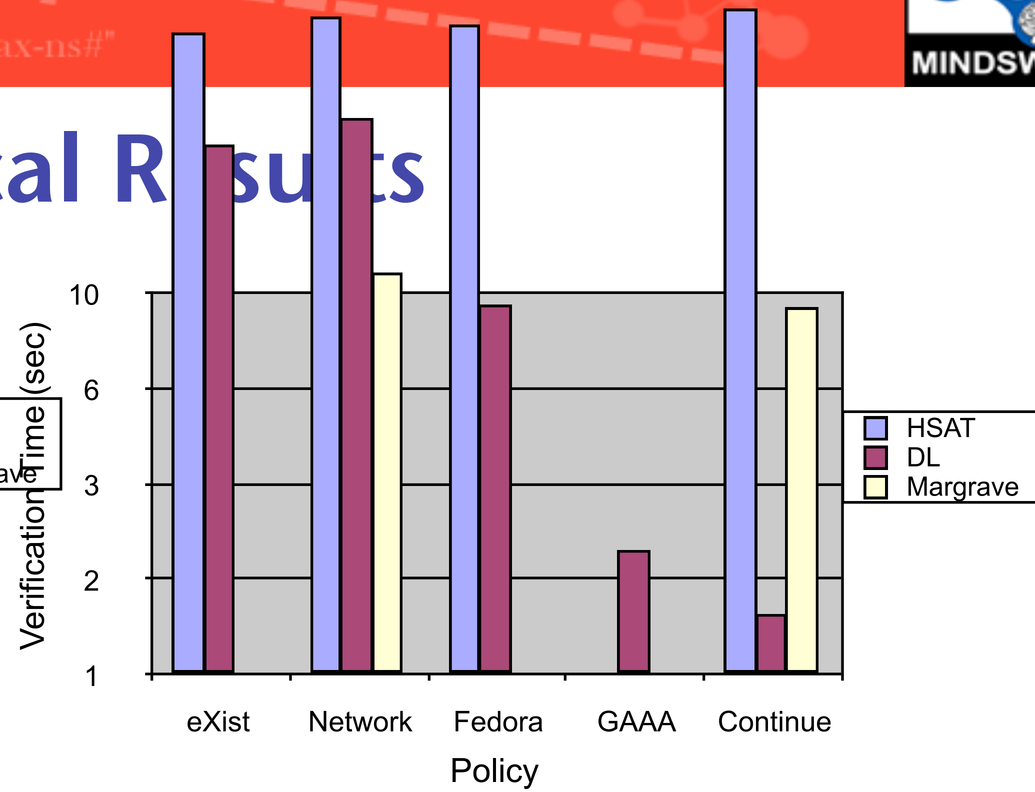
Empirical Results

Two-Part Evaluation

1. Compare against fastest XACML analyzers
   - Margrave (BDD-based), HSAT(SAT-based)
   - Test suite containing real XACML policies
     - Continue, Network, Fedora, GAAA, eXist
     - **Policies selected within expressiveness of HSAT and/or Margrave**
2. **Show approach is practical for expressive, real-world policy use cases**
   - **NASA HQ Data Access Use Case**
   - **HL7 Health care policy**

Empirical Results





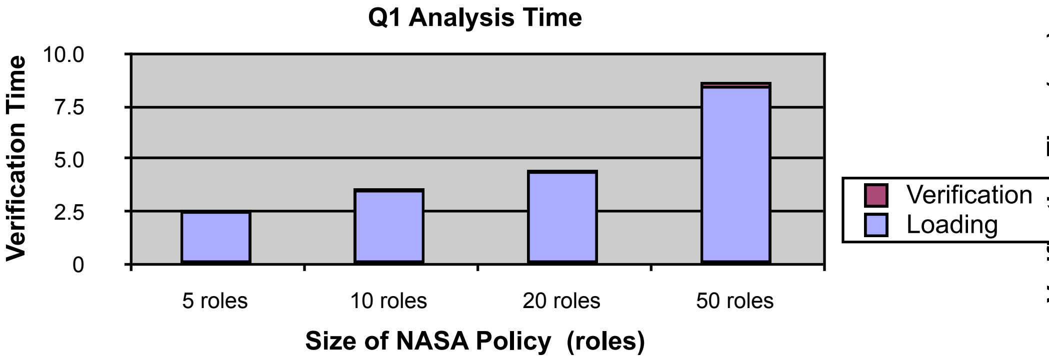
- Tested formal verification and policy comparison
  - Simulated test cases based on policy attributes

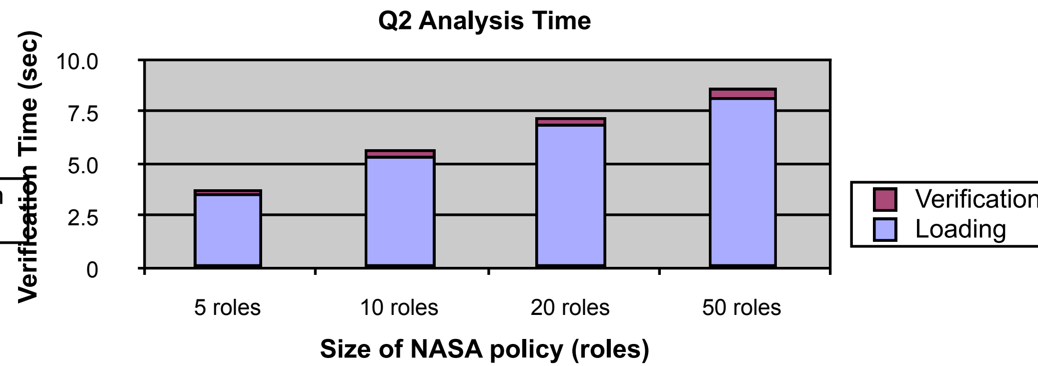
NASA Federated Data Access Use Case

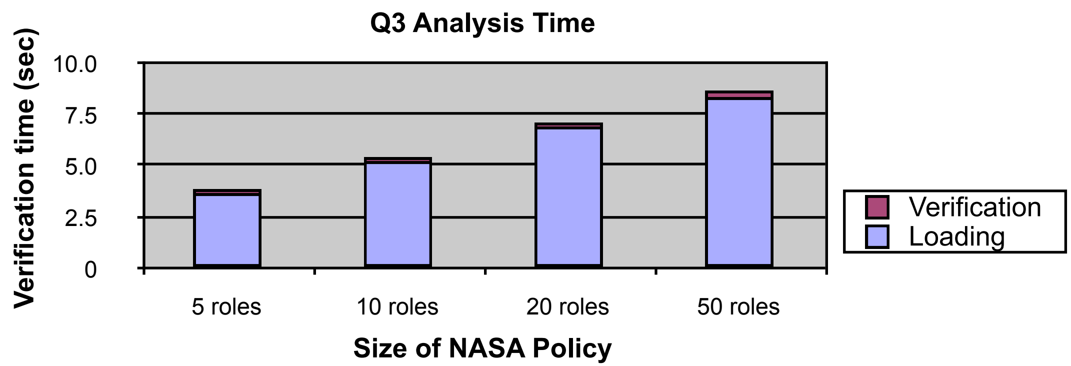
- Collaboration with NASA HQ
  - OWL is already being used at NASA (POPS, BIANCA)
  - NASA interested in XACML+OWL for access control
- Data integration app BIANCA as an example
  - Developed a set of access policies for BIANCA
  - Subjects and resources taken from the NASA Taxonomy
- Resulting XACML policy
  - 4 policy sets (3 departments and 1 general)
    - Each department has 10-15 XACML policies
  - RBAC with data-types and ontology extensions

*For details see: Michael Smith et al. Mother May I? OWL-Based Policy Management at NASA. In Proc. Of the 3rd International Workshop on OWL: Experiences and Directions (OWLED), 2007.

Empirical Results



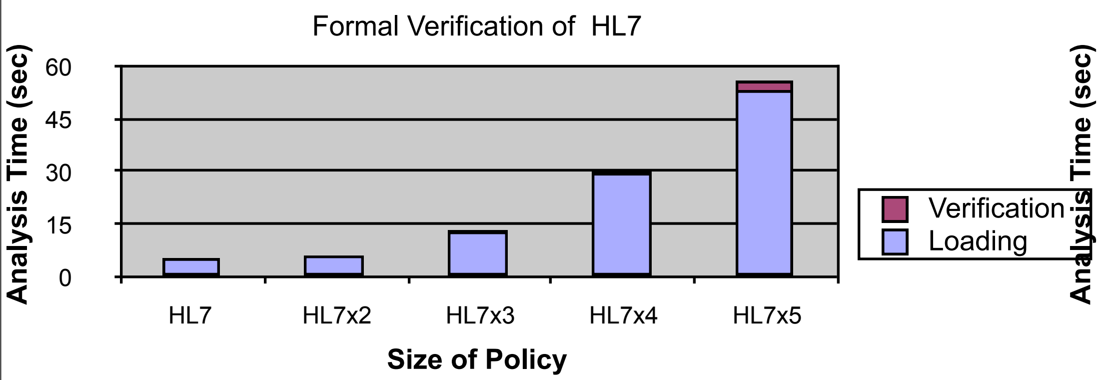


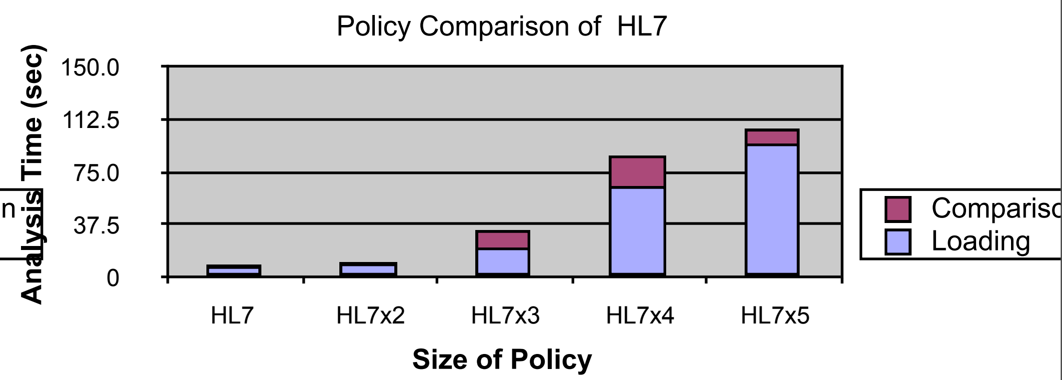


HL7 Healthcare Policy

- Health Level 7 Standard
  - Push towards open standards for electronic health records
  - Access control crucial in this scenario
- Contains a set of RBAC permissions, constraints and scenarios
  - **39 RBAC scenarios represent an instance of a health information system policy**
- HL7 policy
  - Hierarchical RBAC with constraints and data-types
  - Vocabulary domains (role hierarchy in OWL)
- **Converted to HL7 policy to XACML**
  - **107 Policy sets, 100+ attribute values**

Empirical Results





- Tested on original policy and synthetic extensions (more)
- Results demonstrate performance practical for compile-time analysis

Conclusions & Future Work

Contributions

- XACML Semantics and Complexity Results
  - Complexity bounds and comparison of XACML to other logic-based languages
- Developed a static analyzer for XACML policies
  - Demonstrated analyzer is practical for large and expressive policy sets
- Showed framework is applicable to other domains
  - Formalized and analyzed WS-Policy and WS-XACML

Future Work

- Extend reachability analysis
  - Find all minimal reachable sets of policies
- Policy repair service
  - Develop techniques for 'fixing' policies
    - E.g., find a minimal set of constraints to be added s.t. policy satisfies a set of test conditions
- Analyze more expressive policies
  - Obligations and Dynamic Policies
  - Business rules

Questions

Thank You