Semantic Web Policies: Where are we and What is still Missing?

Piero A. Bonatti, Naples University
Daniel Olmedilla, L3S Research Center & Hanover University

November 10th, 2006
SEMANTICS COMPUTER INSTITUTE

Street vendor stalls with various items for sale.
Outline

- Introduction
- Where are we?
- Deployed Application Scenarios
- What is still missing?
- Conclusions
Outline

- Introduction
  - Warming up
  - Some history (from security/trust to knowledge/reasoning)
  - Requirements (expressiveness, user awareness/control)
  - Main challenges

- Where are we?

- Deployed Application Scenarios

- What is still missing?

- Conclusions
Introduction
Why this tutorial?

- Many research papers on policies (also in SW)
- Many approaches (languages and frameworks)
- Little work on comparison, literature review
- Reinventing the wheel
- Can be made more general \(\rightarrow\) greater impact
- Where is the user?
Introduction
About this tutorial (I)

This tutorial is intended to provide
- a basic understanding of requirements of current distributed systems
- a motivation for the use of policies
- a historical review of the field
- an analysis of state of the art

And the most important
- **why should the SW community care**
  - And relevance to rule markup languages
  - open problems and future lines of research
Introduction
About this tutorial (II)

Policies specify the behavior of a system and may be applied to many different areas: security, conversations, business rules, quality of service, etc.

The most common application scenario is security. It covers most of the requirements from other areas.

Although many of our examples and material focus on security, it should be clear all the time that its application is not restricted only to security.
Introduction
About this tutorial (& III)

Slides are wordy so they can be easily understood offline after the tutorial

More definitions and references are available in notes and hidden slides

Tutorial is available from:
http://www.l3s.de/~olmedilla/events/2006/ESWC06/ESWC06_Tutorial.html
WARNING

Or clarification 😊

Ontology ≠ OWL
Introduction

Warming Up: Problems (I)

Institutions and companies need to control the way they

- Make business
- Take decisions
- Offer their assets
- Etc ...

Generally, they need to control how decisions and actions are taken
Policies Are Everywhere

- B2B contracts
  - e.g. quantity flexible contracts, late delivery penalties, etc.
- Negotiation
  - e.g. rules associated with auction mechanisms
- Security
  - e.g. access control policies
- Privacy
  - Information Collection Policies (aka “P3P Privacy Policies”)
  - Obfuscation Policies
- Workflow management
  - What to do under different sets of conditions
- Context aware computing
  - What service to invoke to access a particular contextual attribute
  - Context-sensitive preferences

[ by Norman Sadeh, Semantic Web Policy Workshop panel, ISWC 2005 ]
Introduction

Warming Up: Problems (II)

In the Analog Era, everything is in paper via regulations and written policies/statements but

- They are ambiguous
- Someone has to read them and remember them
- They often change
- Etc...
Introduction
Warming Up: Problems (& III)

In the Digital Era, systems guide many of the decisions and actions to be taken but

- Policies are typically hard-coded
- Policies still change really often
  - Costly process
- Difficult to write policies in a machine-understandable way
  - E.g., try to write a regulation or law in a non-ambiguous way
- Etc ...
Introduction

Warming Up: Challenges

Provide a framework where

- **Behavior is flexible**
  - Can be changed/updated
    - without re-coding, re-compiling, re-installing, etc...
    - In a costless manner

- **Can be managed by administrators/users without needing to be computer experts**

- **Can be understood by normal users**

- **Covers as many different policies as possible**
From security & trust to knowledge and reasoning
From security to KR&R

The security community has already

- Stressed the importance of declarative policy languages
  - To avoid ambiguous or ill-defined policies
  - To separate policies and mechanisms
  - To enable automated policy validation

- Proposed logic-based policy languages
  - To improve readability and maintenance
  - High-level formulation, more natural for untrained user
  - To express / integrate different policies (flexibility)

Languages and standards are starting to be influenced

- Java 2
  - Permissions have a method *implies*
- XACML
  - Built around “rules”
- P3P is a rudimentary ontology
  - Data classes
  - Purpose of use
  - Recipients (immediate and indirect)
- Syntax has a logical flavour
- Semantics is procedural and/or informal
From security to KR&R

Varieties of proposed policy formalisms

- Logic programs
  - With stratified negation as failure
    - Efficient (PTIME)
    - Unambiguous (one canonical model)
  - To make decisions in the absence of explicit information
    - Open and closed policies
  - To support general rules with exceptions
    - Hierarchies of subjects, objects, and actions
  - With periodic temporal expressions
  - With event-condition-action rules

From security to KR&R

Varieties of policy formalisms II

- Deontic logics
  - Permissions, denials and obligations
  - Sometimes in a logic programming fragment
  - Is classical deontic semantics adequate?
    - Start from policies, not from logic

- Description logics
  - Plus rules?
  - Plus nonmonotonic inference?
  - Technical difficulties

From trust management to SW

- Computer security for open systems
  - Occasional users, unknown to the system
    - Traditional authentication is impossible or undesirable
    - Property-based access control
    - Digital credentials

- Privacy issues
  - Unknown servers
    - Limit disclosure of sensitive information
    - Raise the level of trust in the server

- Together security and privacy lead to negotiations
Authentication in open systems
Authentication in open systems
Authentication in open systems

Other password-based systems

- MyProxy
- Kerberos
- Some CAS-based servers
Authentication in open systems
scalability and usability issues

In the absence of more flexible methods

- Web services have to keep accounts for all customers
  - Possibly $>1$ for some customers
  - Some accounts are used very few times
- Users have to create accounts all the time
  - *Many* passwords vs reuse (highly vulnerable)
  - Needs automated password management
- Articulated business policies are discouraged
  - Because they would require continuous user intervention
Authentication in open systems scalability and usability issues

You can get this book:
1. by logging in
2. by supplying an ID and a credit card
3. by providing an Amazon card
Please choose a number or click on a link for more information

Availability: Usually ships within 24 hours. Ships from and sold by Amazon.com.

Only 1 left in stock—order soon (more on the way).

18 used & new available from $26.98

Add to Wish List
Add to Wedding Registry
Authentication in open systems
scalability and usability issues

What one would really want:

- Suppose the *Amazon card* gives you free access to some products
- If you have it, you want to use it automatically
  - Click on the purchase button and that's it
- If you don't, you may want to see something like the next figure
Authentication in open systems scalability and usability issues

WARNING
You are about to pay $10 for paper0123.pdf using your VISA card
Authentication in open systems
scalability and usability issues

Similar desiderata for
ubiquitous/pervasive computing scenarios

- E.g. travellers connect to airport lounge services using
  - Frequent flier cards
  - Pre-paid cards
  - Credit cards
  - Employee credentials (government, airlines, ...)
  - ...

- In a transparent way
  - Well, as far as possible
Beyond authentication
property-based access control

- The amazon card does not necessarily disclose the owner's identity
- Digital credentials can represent also
  - Membership to an association
  - Subscriptions
  - Eligibility to particular services
  - Citizenship, age, and other personal properties
  - Credit cards and other money-related “objects”
  - ...
- Flexible and scalable
  - Domain specific certification authorities
- Privacy preserving
  - Release only what is needed (need-to-know principle)
Privacy issues

- Credentials may be sensitive
  - Credit card numbers, SSN, ...
- Servers cannot be trusted, in general
  - New services, unknown responsibilities, ...
- Credential release may be subject to server certifications
- Seal programs (self regulation): agree to
  - Follow precise practices for protecting information
  - Be subject to audit procedures
  - TRUSTe, BBBOnLine, WebTrust
- Seal program membership can be certified with electronic credentials
Negotiations
symmetric framework: credential are resources

Step 1: Alice requests a service from Bob

Step 2: Bob discloses his policy for the service

Step 3: Alice discloses her policy for VISA

Step 4: Bob discloses his BBB credential

Step 5: Alice discloses her VISA card credential

Step 6: Bob grants access to the service

Expressiveness issues
how to formulate requests

One by one?

- Slow
  - More messages (as opposed to one global request)
- Bad w.r.t. privacy
  - Unnecessary disclosures
  - After submitting $n$ credentials you realize you miss the next
- Example
  - After submitting your $id$ you realize your credit card is not accepted by the server
Expressiveness issues
how to formulate requests

All alternatives at once?

- Less messages (good!)
- Combinatorial explosion:
  - one id and one credit card ➔
    - Passport + VISA
    - Passport + Mastercard
    - ...
    - Student card + VISA
    - Student card + Mastercard
    - ...
    - SSN + VISA
    - SSN + Mastercard
    - ...

P. A. Bonatti, D. Olmedilla
RuleML'06 Tutorial: Semantic Web Policies
Expressiveness issues
how to formulate requests

Send the policy!

- As a compact representation of all alternatives
  - To download paper XY.pdf do one of the following:
    1) Submit an Amazon card
    2) Submit a valid id and an accepted credit card

- The client can
  - Verify that the whole condition can be satisfied
  - Choose the best option
  - Minimizing the sensitivity of disclosed information

- Needs standard rule representation!
Expressiveness issues
how to formulate the policy

- Boolean combinations of credentials
- Restrictions on their attributes
- Possibly **recursive** conditions
  - Credential chains (\(\sim\) transitive closure)
- **A rule-based example:**

  ```prolog
  allow(download(paper1.pdf)) ←
  id(Document),
  Document.name : User,
  credit_card(Card),
  Card.name : User.
  ```
Expressiveness issues
how to formulate the policy

- Boolean combinations of credentials
- Restrictions on their attributes
- Possibly recursive conditions
  - Credential chains (~ transitive closure)
- A rule-based example:

  allow(download(paper1.pdf)) ←
    id(Document),
    Document.name : User,
    credit_card(Card),
    Card.name : User.
Expressiveness issues
how to formulate the policy

- Policies frequently contain concept definitions

allow(download(paper1.pdf)) ←
id(Document),
Document.name : User,
credit_card(Card),
Card.name : User.

Concept id is defined here
id(Document) ←
credential(Document),
Document.type : T,
Document.issuer : CA,
isa(T,id),
trusted_for(CA,id).

More concepts
Therefore policies are

- Knowledge bases
- Containing simple **ontologies**
  - Often *rule-based*
- Shared among peers (during negotiations)
- Enabling interoperability of heterogeneous peers
  - w.r.t. access control and information release
- Policies comprise both
  - Semantic **markup for decision making** and
  - The **ontology** for expressing the markup
Relevance to SW community

Regardless of whether
- Policies protect semantic data
- Policies refer to OWL ontologies

Minimal prerequisites for application: common understanding of
- Logic semantics and rule syntax
- Credential format (X.509 standard)
- No further semantic infrastructure needed
- Lightweight reasoning if Rule-based

Very close to short-term applications
Expressiveness requirements
A broader notion of Policy

The term *policy* covers:

- Security/Privacy policies, Trust management
- Business rules
- Quality of Service directives
- Service-level agreements
  - and more...

They all make decisions based on similar pieces of information (evidence)

- user age,
- nationality,
- customer profile,
- identity,
- reputation...
Examples of policies across business rules and quality of service

- Give customers younger than 26 a 20% discount on international tickets
- Up to 15% of network bandwidth can be reserved by paying with an accepted credit card
- Customers can rent a car if they are 18 or older, and exhibit a driving license and a valid credit card
Context-Sensitive Privacy & Security Policies

Pervasive Computing
- “My colleagues can only see the building I am in and only when they are on company premises”

Enterprise Collaboration
- “Only disclose inventory levels to customers with past due shipments”

DoD Scenarios (e.g. coalition forces)
- “Only disclose ship departure time after the ship has left”
- “Only disclose information specific to the context of ongoing joint operations”

Homeland Security & Privacy (e.g. video surveillance)
- “Only allow for facial recognition when a crime scene is suspected”

[ by Norman Sadeh, Semantic Web Policy Workshop panel, ISWC 2005 ]
Policies are not (only) passive objects

Policies may specify

- Event logging
  - Failed transactions must be logged
  - Log downloads of new articles for one week
- Communications and notifications
  - Notify the administrator about repeated login failures
- Workflow triggering
  - such as (partly) manual registration procedures

i.e. Policies may specify actions

- To be interleaved with the decision process
Strong, Soft, and Lightweight Evidence

How can individuals *prove* their eligibility?

- **Strong evidence**
  - e.g. *digital credentials* (id, credit cards, subscriptions)
- **Soft evidence**
  - e.g. *numerical reputation measures*
  - *PGP, eBay, ...*
- **Lightweight evidence**
  - e.g. "*accept buttons*" (copyright/license agreements)

They should be integrated for balancing:

- **trust level**
- **risk level**
- **computational costs**
- **usability** (fetching credentials, personal assistants)

---

E.g. *micropayments vs. buying plane tickets*
Exploiting “external” systems
or: policies are not islands

Decisions need data, information, and knowledge

- Each organization has its own
  - Already available through legacy software and data
  - A realistic solution must interoperate with them
  - Possible approaches: see logic-based mediators

- Third parties
  - Credit card sites for validity checking
  - Credential repositories

- Variety of web resources
User awareness and control
Widespread security

Most security/privacy violations caused by

- Lack of awareness
  - Users ignore security threats and vulnerabilities
  - Users ignore the policies applied by the systems they use

- Lack of control
  - Users don't know how to personalize their policies

- A social problem
  - Everybody's machine is on the internet
  - Millions of computers can be exploited for attacks

  By taking advantage of the users' lack of technical competence
Widespread security

A recent experiment:
- Several computers connected to the network
  - Different platforms and configurations
- With default policies: intrusion in <5 min.
  - Bias towards functionality
- With personalized policies: safe for 2 weeks
  - Till the end of the experiment

Widespread security

One size does not fit all

- **Strong security policies may cause denial of service**
  - e.g. try to forbid script execution
  - which is one of the most exploited vulnerabilities

Common users are not able to personalize their policies

- **Formulated obscurely**
  - Are *cookies good or bad*?
- Partly cast into program code
Cooperative policy enforcement for occasional users

Crucial for the success of a web service
- Never say (only) “no”!
- Encourage first-time users
- Who don't know how to use your service
- Explain policy decisions
  - Especially failures
  - Advanced queries: Why not
- Guide users in acquiring missing permissions
  - Activate registration workflows
  - Point to credential repositories
  - Advanced queries: How-to, What-if

You can't open this door, but you can ask Alice for permission.
More uses of explanations for policy validation

Post mortem analysis
- How could X get Y?
  - Advanced queries: Why

Static analysis
- Which kind of users can access resource X?
- Which are the permissions of a user with properties XYZ?
  - Advanced queries: How-to, What-if

Denial of service analysis
- Why didn't X get Y?
  - Advanced queries: Why-not
Policies as KBs

One knowledge many uses, e.g.
- Access control
- Communicating requirements
- Explanations
- Validation
- Service selection
  - Use policies as semantic markup
  - Expressing non-functional properties

Different reasoning tasks
- Deduction
- Abduction
- Proof manipulations ...
Main Challenges
Many Policies, One Framework

It is appealing to integrate all policies in one framework

- One common infrastructure
  - for interoperability and decision making
- Where policies can be harmonized & coordinated

Technical challenge

- Harmonize/integrate requirements
  - procedural (ECA) vs. declarative semantics
  - different derivation strategies
  - too complex for one representation language?
Strong, Soft, and Lightweight Evidence

Challenges

- Proper language (discrete + numerical), but
- Reputation models still in early stage
  - new models keep being introduced
  - vulnerabilities (e.g., to coalitions)
- **parametric frameworks?** (current choice of REWERSE)
  - separate reputation module
  - integrated via generic constructs (cf. rule-based mediators)
Interoperability on a larger scale

Challenges

- Different levels of interoperability
  - heterogeneous legacy software and third parties
  - more general credential formats
  - lightweight evidence can be based on any web contents
  - how to explain such requirements in a machine-understandable way?
- a standard semantic web issue – ontologies
- still lightweight?...

Expressive languages, ontology infrastructure

E.g. point to a picture on the conference page to prove you attended ESWC'06

[J. Hendler]
User awareness and control
general challenges

- Explain policies and system decisions
  - Make rules & reasoning intelligible to the common user
  - A classical AI problem – perfectly in line with SW
- Encourage people to personalize their policies
  - Make it easy for users to write their own rules
- Use natural language?
  - “Academic users can download the files in folder historical_data whenever their creation date precedes 1942”
  - Suitably restricted to avoid ambiguities
  - Fortunately, users spontaneously formulate rules
Explanation mechanism
specific challenges

Finding the right tradeoff between

- Quality (2\textsuperscript{nd} generation explanation facilities)
  - Remove irrelevant information
  - User-friendly denotation of internal objects
  - User-oriented description of reasoning

- Framework instantiation effort
  - The framework needs to be adapted to each application domain
  - Expensive in 2\textsuperscript{nd} generation EF (ad hoc KB and engine)
  - Reduce the need for specialized staff
More challenges and more detailed

- Need technical notions
- Some will be tackled in the rest of the tutorial
- From a slightly different perspective, sometimes
Outline

■ Introduction

■ Where are we?
  ■ Requirements for
    ■ Policy Languages
    ■ Policy Frameworks
  ■ Policy Language & Framework State of the Art

■ Deployed Application Scenarios

■ What is still missing?

■ Conclusions
Requirements for Policy Languages
Requirements for Policy Languages
Overview

- Well-defined semantics
- Declarative
- Monotonicity
- Type of Evaluation
- Use of Variables
- Operations/Combinations
- Management of Attribute Credentials
- Delegation of Authority
- After-Disclosure Control
- External functions / Execution of Actions
- Ontology support
- Rule Support
- Protection of policies
- Extensibility
- Lightweight vs. Strong Evidence
- Usability
Requirements for Policy Languages
Well-Defined Semantics

- "No surprises"

- If any party concludes that a policy is satisfied, any other party should conclude the same

- Meaning of policies are independent of the particular implementation

- No space for ambiguity
Requirements for Policy Languages
Declarative

- Closer to the way humans think

- Definition of the what, not the how
  - People do not write algorithms, they write norms
Requirements for Policy Languages
Monotonicity

- Disclosure of additional credentials and policies or execution of actions only results in additional privileges
  - E.g. “grant access if requester is not a student” is invalid

- Only applies to the communication between the client and server
  - Given a VISA, the server may check with a VISA server for the absence of its revocation

- Context (e.g., time, location) is outside of this monotonicity requirement
  - A request made at 16:59 may be successful and the same one be rejected at 17:01

Requirements for Policy Languages
Type of Evaluation

- **Centralized**
  - All information exists locally
  - E.g. Database with permissions or Access Control Lists

- **Distributed Policies, Centralized Evaluation**
  - Policies are distributed
  - Policies are fetched and brought to a central point
  - Reasoning is performed centralized

- **Distributed Evaluation**
  - Policies are distributed
  - Reasoning is distributed
Requirements for Policy Languages
Use of Variables

- Required to
  - Extend semantics (“uncle” or “sameAge” examples)
  - Join different conditions
  - Generalize predicates

Example
- A valid client is such that it has a subscription and such subscription includes the requested object
  \[
  \text{validClient}(\text{Client}, \text{Resource}) \leftarrow \\
  \text{hasSubscription}(\text{Client}, \text{Subscription}), \\
  \text{includes}(\text{Subscription}, \text{Resource})
  \]
- Previous co-authors of a resource’s creator are granted access
  \[
  \text{access}(\text{Document}, \text{Requester}) \leftarrow \\
  \text{isAuthor}(\text{Document.Author, AnyResource}), \\
  \text{isAuthor}(\text{Requester, AnyResource}).
  \]
Requirements for Policy Languages
Operations / Combinations

- Operations
  - Nested policies need to be combined
  - Disjunction, conjunction, negation, xor, etc.

Example

- Access granted to
  employees
  OR
  students AND student is European citizen
  OR
  clients AND client is not blacklisted

Requirements for Policy Languages
Management of Attribute Credentials

- Disclosed credentials need to be accessed
- Their properties may be the base for a decision

Example:
- Grant access if the credential is issued by “University of Hannover” AND has type “student credential”
Requirements for Policy Languages
Delegation of Authority

- Decisions are not always local
  - Policies used during evaluation may be distributed
  - Fetching and centralized evaluation may not be possible due to privacy concerns
- Required to delegate decisions to other (possibly external) entities

Example:
- Access is granted if my partner company says so
- A credit card is accepted if VISA says it is valid
Requirements for Policy Languages
After-Disclosure Control

- Parties disclose information only if the requester party is entitled to receive it
- However, once information is disclosed, control over it is lost

- So far, only voluntary compliance is possible, not enforceable

- Needed to control information after its disclosure
  - The information I disclose to you cannot be disclosed to 3rd parties
  - You can give my e-mail only to your friends (one step forward) but no more
Requirements for Policy Languages
External functions / Execution of Actions

- Unfeasible to have a single system with all institution information (e.g. legacy systems)
  - Duplication is undesirable

- Policies may involve the execution of actions outside the policy framework
  - Log each new request
  - If the negotiation succeeds, send a notification e-mail

- It should be possible to specify properties for the action, e.g., the actor that must execute the action
  - E.g. Credential fetching
Requirements for Policy Languages
Ontology Support

- Different entities may have different definitions

- Interoperability
  - Needed to “explain” what a concept means#
  - Sometimes difficult only with rules
  - Other paradigms may need to be integrated

- Definition of concepts using Ontologies
  - E.g. type of credentials
    - Disclose a credential of type credit card. Credit cards are VISA, Master Card and AmEx
Requirements for Policy Languages
Rule Support

- People tend to write policies as rules
  - Declarative
  - Event Condition Action Rules
  - Rules are intuitive and natural way of thinking

- Policies are used as examples in the W3C Rule Interchange Format (RIF) working group
Requirements for Policy Languages
Protection of Policies

- Policies may be sensitive
  - Access allowed only to Sun or Microsoft employees
  - Medical record can be retrieved by the patient or his psychiatrist
  - Police file accessible only by his parole officer
  - My pictures only available to my friends

- In this case, policies are hidden till later stages where more information is available
- Process is not a 1-step communication anymore
- Now it is a negotiation
Requirements for Policy Languages

Extensibility

- Requirements evolve over time
- The language should be able to adapt to new requirements

Extensible to new
- Operators
- Constructors
- Definitions
- Concepts
Requirements for Policy Languages
Lightweight vs. Strong Evidence

Policies may need to distinguish on whether information provided as been signed or not

- Lightweight
  - Forms (e.g. user and password, license acceptance)
- Strong / Signed
  - Credentials

Example
- Log in with a user/password
- Access granted if credit card is provided
### Requirements for Policy Languages

**Strong Evidence: Standard Certificates**

<table>
<thead>
<tr>
<th>X.509 Certificate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subject</strong></td>
</tr>
<tr>
<td><strong>Issuer</strong></td>
</tr>
<tr>
<td><strong>Period of Validity</strong></td>
</tr>
<tr>
<td><strong>Other Info</strong></td>
</tr>
</tbody>
</table>

- **Possibility for additional information via extensions**
- **Type of extensions**
  - Critical
    - Credential should be discarded if the extension is not understood
  - Non-Critical
- **Here ontologies come into play...**
Requirements for Policy Languages
Usability: Example Policy in Cassandra

loc@iss.canActivateRole(adm,NHS-Caldicott-guardian-cert(org,cg,start,end))
  ←
  loc@iss.hasActivatedRole(adm, RA-admin()),
  loc@iss.hasActivatedRole(x, NHS-health-org-cert(org, start01, end01)),
  %start in [start01, end01], end in [start01, end01], start < end,
  loc='RA-East', iss='RA-East'%

loc@iss.canDeactivate(adm,x,NHS-Caldicott-guardian-cert(org,cg,start,end))
  ←
  loc@iss.hasActivatedRole(adm, RA-admin()),
  %loc='RA-East', iss='RA-East'%

loc@iss.other-NHS-health-org-regs(count<y>, x, org, start, end)
  ←
  loc@iss.hasActivatedRole(y, NHS-health-org-cert(org, start01, end01)),
  %start in [start01, end01], end in [start01, end01], start<end,
  x != y or start != start01 or end != end01,
  loc='RA-East', iss='RA-East'%
Requirements for Policy Languages
Usability: is atom annotation good?

loc@iss.canActivateRole(adm,NHS-Caldicott-guardian-cert(org,cg,start,end))
←
loc@iss.hasActivatedRole(adm, RA-admin()),
loc@iss.hasActivatedRole(x, NHS-health-org-cert(org, start01, end01)),
%start in [start01, end01], end in [start01, end01], start < end,
loc='RA-East', iss='RA-East'%

loc@iss.canDeactivate(adm,x,NHS-Caldicott-guardian-cert(org,cg,start,end))
←
loc@iss.hasActivatedRole(adm, RA-admin()),
%loc='RA-East', iss='RA-East'%

Some annotation should be consistent across all rules
■ Replication, redundancy
■ Correctness is on the shoulders of policy writers
“Too often, only the PhD student that designed a policy language or framework can use it effectively”

[ by Kent E. Seamons, Semantic Web Policy Workshop panel, ISWC 2005 ]
Requirements for Policy Frameworks
Requirements for Policy Frameworks
Overview

- Conflict resolution / combination of policies
- Accountability / Proofs
- Implementation
- Tools / applications
- Support Explanations
Requirements for Policy Frameworks
Conflict Resolution

- Is this expressiveness needed?
  - Depending on scenarios it may not
  - Guarantee must exist that every conflict will be detected

- Given a request, different policies may apply
- Results of conflict evaluation may be conflicting

- Resolution mechanism should be provided

Example:
- A policy grants access and another denies it
- Obligation to do something but prohibited to do it
Requirements for Policy Frameworks
Accountability/Proofs

- Access control decisions may be performed in different entities than the ones holding the resources
- It should be possible to proof the result of an access control decision (e.g., negotiation) to third parties
- Proof-carrying code + credentials allow that
Requirements for Policy Frameworks Implementation

- Obvious, isn’t it?

- Unfortunately, for many policy languages there is no implementation, it is only a prototype and/or is not available for general use, e.g.:
  - REI: needs old XSB and obsolete libraries
  - Ponder: not available anymore (announcement for Ponder2)
  - KAOS: under request one gets access to a client to test the GUI and basic reasoning
  - Cassandra: not accessible
  - PeerTrust, Protune: proofs of concept

- If no well-defined semantics, implementations may differ
  - Space for ambiguities
Requirements for Policy Frameworks
Tools / Applications

- Templates / Profiles
  - Do not replace user-friendly languages

- Editors

- Validation / Verification

- Explanations

...
Policy Language/Framework State of the Art
Where are we? Classification

- **Well-defined Semantics**
  - RBAC
  - Kaos

- **No Formal Semantics**
  - ACL
  - Java Policies
  - Ponder
  - XACML
  - P3P

- **Distributed Policies, Centralized Evaluation**
- **Distributed Evaluation**

- **Centralized Evaluation**
- **Distributed Policies, Centralized Evaluation**
- **Distributed Evaluation**

- PSPL
  - SD3, RT
  - PeerTrust
  - Cassandra
  - Protune
  - PeerAccess
XACML Overview (I)

- `<Rule>`, `<Policy>` and `<PolicySet>`
  - `<Rule>`
    - boolean expression
    - Applicable according to `<Target>` & `<Condition>`. `<Effect>` only Permit or Deny
    - not accessible by PDP
  - `<Policy>`
    - set of `<Rule>` and procedure for its combination
    - Basic unit used by the PDP
    - May have obligations attached
  - `<PolicySet>`
    - Set of `<Policy>` or `<PolicySet>` and procedure for its combination
    - Combine separate policies into a single combined policy

- Combining algorithms
  - Denyoverrides (conjunction), Permitoverrides (disjunction), First-applicable, Only-one-applicable
  - Extensible

- Multiple subjects in different capacities (attrib. subject-category)
XACML
Overview (& II)

- Attributes of the subject & object
  - `<SubjectAttributeDesignator>` or `<AttributeSelector>` (in the context)
  - `<ResourceAttributeDesignator>` or `<AttributeSelector>` (in the context)
- Multi-valued attributes
- Content of an information resource (only if document is in XML)
  - `<AttributeSelector>` (in the context)
- Mathematical operators on attributes (<Apply FunctionId=""/>)
  - Arithmetic, set operators, boolean, equality and comparison
  - Extensible
- Abstract the location and retrieval of policies but handle distributed sets of policies
  - Check with `<Target>` if the policy is applicable or not
  - However, they must be retrieved to a central place for evaluation
- Rapidly identify applicable policies (using `<Target/>`)
- Set of actions to be executed
  - In conjunction with policy evaluation `<Obligations>`

[ OASIS eXtensible Access Control Markup Language (XACML) 2.0
**XACML**

**Data Flow Diagram**

1. **PAP**
   - Policy

2. **Access Request**
   - PEP

3. **Request**
   - Context Handler

4. **Request Notification**
   - PDP

5. **Attributes Queries**
   - Context Handler

6. **Attribute Query**
   - Context Handler

7. **Attributes**
   - PAP

8. **Attribute**
   - PEP

9. **Resource Content**
   - Resource

10. **Attributes**
    - Context Handler

11. **Response Context**
    - Context Handler

12. **Response**
    - PEP

13. **Obligations**
    - Obligations Service

**Centralized Point of Evaluation**
**XACML Example**

```
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
  http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
  PolicyId="urn:oasis:names:tc:example:SimplePolicy1"
  RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides">
  <Description>Medi Corp access control policy</Description>

  <Target/>
  <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:SimpleRule1"
    Effect="Permit">
    <Description>Any subject with an e-mail name in the med.example.com domain can
    perform any action on any resource.</Description>
    <Target><Subjects><Subject>
      <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
          med.example.com
        </AttributeValue>
        <SubjectAttributeDesignator
          AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
          DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
      </SubjectMatch>
    </Subject>
  </Target>
</Rule>
</Policy>
```
XACML
Analysis of the Language (I)

- Well defined semantics
  - Procedural semantics, in Haskell (functional programming language)
- Declarative
  - No
- Monotonicity (respect to policies, credentials and actions)
  - There is no negation. Combination with “first-applicable” makes it too procedural
- Type of Evaluation
  - Distributed Policies, centralized evaluation
- Use of Variables
  - Implicit for Subject, Action, Resource, Environment and their attributes
- Operations/Combinations (conjunction, disjunction, negation, xor, etc.)
  - Conjunction, disjunction, first-applicable, only-one-applicable
  - Extra operators may be defined
- Management of Attribute Credentials
  - Yes, if passed in the context
- Delegation of Authority
  - No
XACML

Analysis of the Language (& II)

- After-Disclosure Control
  - No
- External functions / execution of actions
  - Obligations. Only deferred ones
- Ontology support
  - No
- Rule Support
  - Rules without variables. Nested rules allowed bound by Subject, Action, Resource & Environment attributes only.
  - Not possible to chain rules
- Protection of policies
  - No. Retrieval of applicable policies and centralized point of evaluation
- Extensibility
  - Yes. New algorithms for combination and operators
- Lightweight vs. Strong Evidence
  - Not explicitly
- Usability
  - Difficult with XML syntax. Relatively good for simple policies (if using tools) but difficult if they become complex
XACML
Analysis of the Framework

- Conflict resolution / combination of policies
  - Deny overrides, Permit overrides, first-applicable, only-one-applicable
- Accountability / Proof carrying code
  - No
- Implementation
  - Yes
- Tools / applications
  - Parthenon XACML Evaluation Engine, Sun's XACML Open Source, XACML.NET, UMU XACML editor, AXESCON XACML 2.0 Engine
- Support Explanations
  - No
P3P
Overview (I)

- Platform for Privacy Preferences
  - Standard XML-format with common vocabulary
    - It is a schema, not a language
  - Policies are fetched from the Website being accessed
  - Support automatic analysis of privacy statements
    - According to user preferences (e.g., using APPEL)
  - It does not enforce compliance
P3P
Syntax (I)

ADOR:
Rules:
Tutorial:
Semantic Web Policies

<Policy>
- Includes
  - one or more statements
  - Name and URI to the natural language policy

<Entity>
- Describes the legal entity stating the privacy practices

<Access>
- Indicates whether gathered data can be accessed after it has been collected

<Disputes>
- Describe the dispute resolution procedure in case of possible conflicts over the policy
- Enterprise is still liable according to normal law procedures

[ Platform for Privacy Preferences (P3P) 1.0 http://www.w3.org/P3P ]
P3P
Syntax (& II)

<Statement>
- Describe data practices applied to data collected
- <Non-Identifiable>
  - No data collected or properly anonymized
- <Purpose>
  - Purpose of the collection of data
  - E.g., <current/>, <develop/>, <telemarketing/>, etc.
- <Recipient>
  - Which entities may access the data
  - E.g., <ours>, <public>, etc.
- <Retention>
  - How long is the data going to be stored
- <Data-group>
  - Type of data the site collects
  - E.g., #user.home-info.city, #user.login.id, #user.gender, etc.
  - <Category>
    - Classification of data elements to ease user preferences
    - E.g., <financial/>, <navigation/>, <state/>, etc.
P3P
Example

<POLICIES xmlns="http://www.w3.org/2002/01/P3Pv1">
   <POLICY name="forBrowsers"
      discuri="http://www.catalog.example.com/PrivacyPracticeBrowsing.html" xml:lang="en">
      <ENTITY>
         <DATA-GROUP>
            <DATA ref="#business.name">CatalogExample</DATA>
            <DATA ref="#business.contact-info.postal.street">4000 Lincoln Ave.</DATA>
            <DATA ref="#business.contact-info.postal.city">Birmingham</DATA>
            <DATA ref="#business.contact-info.postal.postalcode">48009</DATA>
            <DATA ref="#business.contact-info.postal.country">USA</DATA>
            <DATA ref="#business.contact-info.online.email">catalog@example.com</DATA>
         </DATA-GROUP></ENTITY>
      </POLICY>
      <ACCESS><nonident/></ACCESS>
      <DISPUTES-GROUP>
         <DISPUTES resolution-type="independent" service="http://www.PrivacySeal.example.org"
            short-description="PrivacySeal.example.org">
            <REMEDIES><correct/></REMEDIES>
         </DISPUTES-GROUP>
      </POLICY>
      <STATEMENT>
         <PURPOSE><admin/><develop/></PURPOSE>
         <RECIPIENT><ours/></RECIPIENT>
         <RETENTION><stated-purpose/></RETENTION>
         <DATA-GROUP>
            <DATA ref="#dynamic.clickstream"/>
            <DATA ref="#dynamic.http"/>
         </DATA-GROUP>
      </STATEMENT>
   </POLICY>
</POLICIES>
P3P
You are probably already using it

The page you are looking for is currently unavailable. The site might be experiencing technical difficulties, so please try the following:

- Click the Refresh button, or try again later.
- If you typed the page address in the Address box, make sure that it is spelled correctly.
- To check your connection settings, click the Tools menu, click Internet Options, and then click Settings. The settings should now be displayed. For example, the settings provided by your local area network (LAN) or Internet service provider (ISP).
- See if your Internet connection settings are available in Internet Explorer. You can set Microsoft Windows to examine your network connection and automatically discover network settings, if your network administrator has enabled this feature. For example, in Internet Explorer, click Tools, and then click Internet Options. On the Connections tab, click LAN Settings, and then click Automatic connection settings. Click OK.
- Some sites require 128-bit connection security. Check your network settings, and then click OK.

You can also prevent most pop-up windows from appearing by clicking the Settings button in the Internet Options dialog box, clicking the Internet tab, and then clicking Block pop-ups.
P3P
Analysis of the Language (I)

- Well defined semantics
  - No. Policies may even be ambiguous
  - From the spec: “In cases where the P3P vocabulary is not precise enough, sites should use the vocabulary terms that most closely match their practices and provide further explanations”

- Declarative
  - It does not apply

- Monotonicity (respect to policies, credentials and actions)
  - It does not apply

- Type of Evaluation
  - Centralized. Fetching of the applicable policy and matching against preferences

- Use of Variables
  - It does not apply

- Operations/Combinations (conjunction, disjunction, negation, xor, etc.)
  - No. Only one policy applies for each URI

- Management of Attribute Credentials
  - No

- Delegation of Authority
  - No
P3P
Analysis of the Language (& II)

- After-Disclosure Control
  - No
- External functions / execution of actions
  - No
- Ontology support
  - No. Common vocabulary
- Rule Support
  - No
- Protection of policies
  - No. Policies are public
- Extensibility
  - Yes. Extension to the syntax via <Extension>
- Lightweight vs. Strong Evidence
  - It does not apply
- Usability
  - Simple schema with predefined vocabulary
P3P
Analysis of the Framework

- Conflict resolution / combination of policies
  - No
- Accountability / Proof carrying code
  - No
- Implementation
  - Yes. Integrated in Internet Explorer
- Tools / applications
  - No
- Support Explanations
  - No
Kaos
Overview

- Framework for specification, management, conflict resolution and enforcement of policies
- Uses OWL ontologies
- Policies may be
  - Positive authorization: permits execution of an action
  - Negative authorization: forbids execution of an action
  - Positive obligation: require execution of an action
  - Negative obligation: waive from execution of an action

- Policies are represented as instances of the appropriate type of policy

[ Uszok, Bradshaw, Jeffers, Suri, Hayes, Breedy, Bunch, Johnson, Kulkarni, Lott. KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In POLICY, page 93, 2003. ]
<owl:Class rdf:ID="RetrieveFileAction">  
  <owl:intersectionOf>  
    <owl:Class rdf:about="#AccessAction"/>  
    <owl:Class><owl:Restriction>  
      <owl:onProperty rdf:resource="#performedBy"/>  
      <owl:someValuesFrom>  
        <owl:Class>  
          <owl:oneOf rdf:parseType="Collection">  
            <owl:Thing rdf:about="#EmployeeInstitutionXYZ"/>  
          </owl:oneOf>  
        </owl:Class>  
      </owl:someValuesFrom>  
    </owl:Restriction></owl:Class>  
  </owl:intersectionOf>  
</owl:Class>

  <policy:controls rdf:resource="#RetrieveFileAction"/>  
  <policy:hasPriority>1</policy:hasPriority>  
</policy:PosAuthorizationPolicy>
Kaos
Reasoning

Uses DL subsumption mechanisms to reason over policies

- Check for applicable policy
  - All policies whose controlled actions can be performed by a class or instance of an actor
  - Check if an action instance is an instance of some action class controlled by existing policies

- Detect policy conflicts
  - Check if 2 subclasses of an action controlled by two selected policies are disjoint
  - Check if the subclass of an action controlled by a policy with lower priority is a subclass of the action controlled by the policy with higher priority
Kaos
Policy Conflicts

Types
- Positive vs. negative authorization
- Positive vs. negative obligation
- Positive obligation vs. negative authorization

Static Conflict Resolution Algorithm
- Policy Harmonization
- Automatic
- At design time
- According to policy precedence conditions
KAOS
Analysis of the Language (I)

- Well defined semantics
  - Yes. Based on DL
- Declarative
  - Yes
- Monotonicity (respect to policies, credentials and actions)
  - It does not have negation
- Type of Evaluation
  - Policies are delivered to agents and evaluation is centralized.
- Use of Variables
  - No
- Operations/Combinations (conjunction, disjunction, negation, xor, etc.)
  - No
- Management of Attribute Credentials
  - No
- Delegation of Authority
  - No
KAOS
Analysis of the Language (& II)

- After-Disclosure Control
  - No
- External functions / execution of actions
  - No
- Ontology support
  - Yes. OWL ontologies
- Rule Support
  - No
- Protection of policies
  - No
- Extensibility
  - Yes. Via ontologies
- Lightweight vs. Strong Evidence
  - No
- Usability
  - Logic language (DL). Administration tools exist
KAOS
Analysis of the Framework

- Conflict resolution / combination of policies
  - Yes. Automatic algorithm at design time
- Accountability / Proof carrying code
  - No
- Implementation
  - Yes
- Tools / applications
  - Administration tool (KPAT)
  - Enforcers to ensure compliance with policies
- Support Explanations
  - No
REI 2.0
Overview (I)

- Policies as norms of behavior
- Expressed in OWL-Lite
- Includes logic-like variables

- A policy is a list of rules and a context used to define the policy domain

- `<policy:context>`
  - Conditions over attributes of entities

- `<policy:grants>`
  - Associate deontic object with a policy

REI 2.0
Overview (& II)

- Expresses policies according to deontic concepts
  - Permission
  - Prohibition
  - Obligation
  - Dispensation

- Uses speech acts to decentralized control
  - Delegation & revocation of permissions
  - Request & cancellation of actions
REI 2.0
Metapolicies

Defaults

- Behavior
  - Permitted by default, prohibited by default, explicit statement required
- MetaDefault: which metapolicy is invoked first
  - Check modality first or check priority first

Conflict Resolution

- Conflict of Modality
  - Right and prohibition
  - Obligation and dispensation
- Conflict of Obligation and Prohibition
- Priorities
  - A1 is given higher priority than B1 where A1 can be rule or policy
    - E.g., school policy overrides department policy)
- Precedence
  - Positive: permission and obligation override the others
  - Negative: prohibition and dispensation override the others
<policy:Policy rdf:ID="CSDeptPolicy">
  <policy:context rdf:resource="#IsMemberOfCS"/>
  <policy:grants rdf:resource="#Granting_StudentLaserPrinting"/>
  <policy:defaultBehavior rdf:resource="ExplicitPermExplicitProh"/>
  <policy:defaultModality rdf:resource="PositiveModalityPrecedence"/>
  <policy:metaDefault rdf:resource="CheckModalityPrecFirst"/>
</policy:Policy>

<constraint:SimpleConstraint rdf:ID="IsMemberOfCS">
  <constraint:subject rdf:resource="#PersonVar"/>
  <constraint:predicate rdf:resource="#univ:affiliation"/>
  <constraint:object rdf:resource="#univ;CSDept"/>
</constraint:SimpleConstraint>

<policy:Granting rdf:ID="Granting_StudentLaserPrinting">
  <policy:to rdf:resource="#PersonVar"/>
  <policy:deontic rdf:resource="#Perm_StudentPrinting"/>
  <policy:requirement rdf:resource="#IsLaserPrinterAndPhStudent"/>
</policy:Granting>

<deontic:Permission rdf:ID="Perm_StudentPrinting">
  <deontic:actor rdf:resource="#PersonVar"/>
  <deontic:action rdf:resource="#ObjVar"/>
  <deontic:constraint rdf:resource="#IsStudentAndBWPrinter"/>
</deontic:Permission>
REI 2.0
Analysis of the Language (I)

- Well defined semantics
  - Yes?
- Declarative
  - Yes
- Monotonicity (respect to policies, credentials and actions)
  - Yes. It does not model credentials or action execution
- Type of Evaluation
  - Fetching of relevant policies and centralized evaluation
- Use of Variables
  - Yes
- Operations/Combinations (conjunction, disjunction, negation, xor, etc.)
  - Conjunction, disjunction, negation as failure
- Management of Attribute Credentials
  - No
- Delegation of Authority
  - No
REI 2.0
Analysis of the Language (& II)

- After-Disclosure Control
  - No
- External functions / execution of actions
  - No
- Ontology support
  - Yes. OWL ontologies
- Rule Support
  - No
- Protection of policies
  - No
- Extensibility
  - Yes. Via ontologies
- Lightweight vs. Strong Evidence
  - No
- Usability
  - ?
REI 2.0
Analysis of the Framework

- Conflict resolution / combination of policies
  - Yes. Based on priorities and metapolicies

- Accountability / Proof carrying code
  - No

- Implementation
  - Yes. Using Flora and F-OWL

- Tools / applications
  - Specification editor is on-going
  - What-if analysis

- Support Explanations
  - No
RT

Overview

- Set of role based trust management languages
  - $RT_0$, $RT_1$, $RT_2$, $RT^T$, $RT^D$

- Combines RBAC, trust management and delegation logic

RT

RT$_1$ credentials

- Simple member
  - A.R $\leftrightarrow$ D
  - isMember (D, A.R)

- Simple containment
  - A.R $\leftrightarrow$ B.R$_1$
  - isMember (?z, A.R) $\leftrightarrow$ isMember (?z, B.R$_1$)

- Linking containment
  - A.R $\leftrightarrow$ B.R$_1$.R$_2$
  - isMember (?z, A.R) $\leftrightarrow$ isMember (?x, B.R$_1$), isMember (?z, ?x.R$_2$)

- Intersection containment
  - A.R $\leftrightarrow$ B$_1$.R$_1$ $\cap$ ... $\cap$ B$_k$.R$_k$
  - isMember (?z, A.R) $\leftrightarrow$
    - isMember (?z, B$_1$.R$_1$), ..., isMember (?z, B$_k$.R$_k$)
Example

EPub. preferred ® EOrg. preferred
EOrg. preferred ® IEEE. member
EPub. student ® EPub. university. stuID
EPub. university ® ABU. accredited
ABU. accredited ® StateU
StateU. stuID ® Alice
IEEE. member ® Alice
RT

Analysis of the Language (I)

- Well defined semantics
  - Yes
- Declarative
  - Yes
- Monotonicity (respect to policies, credentials and actions)
  - There is no negation
- Type of Evaluation
  - Distributed Policies, Centralized Evaluation
- Use of Variables
  - Implicit variables
- Operations/Combinations (conjunction, disjunction, negation, xor, etc.)
  - Intersection, union, product containment, exclusive product containment
  - Extensible
- Management of Attribute Credentials
  - Yes
- Delegation of Authority
  - Yes
Analysis of the Language (& II)

- After-Disclosure Control
  - No
- External functions / execution of actions
  - No
- Ontology support
  - No
- Rule Support
  - Rules with implicit variables
- Protection of policies
  - No
- Extensibility
  - Yes
- Lightweight vs. Strong Evidence
  - No
- Usability
  - Logic language
Analysis of the Framework

- Conflict resolution / combination of policies
  - Does not apply
- Accountability / Proof carrying code
  - No
- Implementation
  - Yes
- Tools / applications
  - Not known
- Support Explanations
  - No
PeerTrust
Overview (I)

- Based on guarded distributed logic programs
- Distributed evaluation of policies

Definite Horn Clauses of the form

$$\text{lit}_0 \leftarrow \text{lit}_1, \ldots, \text{lit}_n$$

References to other peers

- $$\text{lit}_i \; @ \; \text{Issuer}$$
- $$\text{lit}_i \; \$ \; \text{Requester}$$

Signed Rules

- student(alice) @ uiuc signedBy [uiuc]

Guards: specify a partial evaluation order for the literals

- request(Course, Session) $ \; \text{Requester} \leftarrow $
  - drivingLicense(Requester) @ caState @ Requester
  - getCourse(Course, Session).

[Gavriloaie, Nejdl, Olmedilla, Seamons, Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. European Semantic Web Symposium (ESWS 2004)]
PeerTrust
Overview (& II)

- Distributed policy evaluation
  - Delegation of authority provokes evaluation on different peers
  - E.g., ask my partner if requester is a valid client

- Policy protection
  - Policies protected by policies
  - Sensitive policies are disclosed after required level of trust is established
  - Negotiations

- Signing statements
  - Explicitly represented in the policies
  - Modelling of strong evidence vs. no evidence

- Distributed proofs
  - Constructed during policy evaluation
validClient (User) ←
validClient(User) @ ‘Partner Company A’.

freeEnroll(Course, Requester) $ Requester ←
policeOfficer(Requester) @ ‘California State Police’ @ Requester,
rdfType(Course, ‘http://.../elena#Course’),
dcLanguage(Course, ‘es’),
creditUnits(Course, X),
X <= 1.

policeOfficer(‘Alice Smith’) @ ‘California State Police’ $ Requester ←
member(Requester) @ ‘Better Business Bureau’ @ Requester
| signedBy ['California State Police'].
PeerTrust
Analysis of the Language (I)

- Well defined semantics
  - Yes
- Declarative
  - Yes
- Monotonicity (respect to policies, credentials and actions)
  - There is no negation
- Type of Evaluation
  - Distributed
- Use of Variables
  - Yes
- Operations/Combinations (conjunction, disjunction, negation, xor, etc.)
  - Conjunction, Disjunction
- Management of Attribute Credentials
  - Yes
- Delegation of Authority
  - Yes
PeerTrust
Analysis of the Language (& II)

- After-Disclosure Control
  - Yes, restrictive via contexts
- External functions / execution of actions
  - No
- Ontology support
  - Import mechanism for RDF data
- Rule Support
  - Yes
- Protection of policies
  - Yes
- Extensibility
  - Yes, via libraries
- Lightweight vs. Strong Evidence
  - Yes. An extension defines ‘@’ as lightweight evidence and ‘@@’ as strong evidence. Also, signed rules exist
- Usability
  - Logic language
PeerTrust
Analysis of the Framework

- Conflict resolution / combination of policies
  - Does not apply
- Accountability / Proof carrying code
  - Yes
- Implementation
  - Yes. Deployable in a jar file (e.g., in an applet)
- Tools / applications
  - Protégé and RCP Editors, Integration into Web servers and Grid environments
- Support Explanations
  - No
Protune
Specification

**PRovisional TrUst NEgotiation framework**

- Supports general provisional-style actions
- An extendible declarative metalanguage for driving decisions and extensibility
- A parameterized negotiation procedure, that gives a semantics to the metalanguage
  - Policy Filtering
- **Integrity constraints** for negotiation monitoring and disclosure control.
- General, **ontology**-based techniques for importing and exporting metapolicies and for smoothly integrating language extensions.

[Bonatti, Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. IEEE POLICY 2005]
Protune
Specification

Based on normal logic program $A \leftarrow L_1, \ldots, L_n$

Categories of predicates are

- **Decision Predicates:**
  - **Allow():** queried by the negotiation for access control decisions
  - **Sign():** used to issue statements signed by the principal owning the policy

- **Abbreviation/Abstraction Predicates**

- **Constraint Predicates:** comprise usual equality and disequality predicates

- **State Predicates:** decisions according the state
  - **State Query Predicates:** read the state without modifying it
  - **Provisional Predicates:** may be made true by means of associated actions that may modify the current state
    - E.g. credential(C,K), declaration(), logged(X,logfile_name)

[Bonatti, Olmedilla. Driving and Monitoring Provisional Trust Negotiation with Metapolicies. IEEE Policies for Distributed Systems and Networks (POLICY 2005)]
## Protune Metapolicies

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>provisional predicates</td>
<td>commands</td>
</tr>
<tr>
<td>actor</td>
<td>provisional predicates</td>
<td>self, peer</td>
</tr>
<tr>
<td>aggregation_method</td>
<td>cost and sensitivity attributes</td>
<td>max, min, sum, adopt(Predicate)</td>
</tr>
<tr>
<td>cost</td>
<td>provisional predicates</td>
<td>number</td>
</tr>
<tr>
<td>evaluation</td>
<td>state predicates</td>
<td>immediate, delayed, concurrent</td>
</tr>
<tr>
<td>expected_outcome</td>
<td>provisional predicates</td>
<td>success, failure, undefined, unknown</td>
</tr>
<tr>
<td>explanation</td>
<td>literals and rules</td>
<td>string expression</td>
</tr>
<tr>
<td>ontology</td>
<td>abbreviation predicates, credentials, declarations, actions</td>
<td>URI</td>
</tr>
<tr>
<td>predicate</td>
<td>literals</td>
<td>predicate names</td>
</tr>
<tr>
<td>selection_method</td>
<td>negotiator</td>
<td>certain_first, order(attribute_list), adopt(Predicate)</td>
</tr>
<tr>
<td>sensitivity</td>
<td>predicates, literals, rules</td>
<td>public, private, not_applicable</td>
</tr>
<tr>
<td>type</td>
<td>predicates, literals</td>
<td>abbreviation, constraint, decision, state_predicate, provisional, state_query</td>
</tr>
</tbody>
</table>
Protune

Examples of metapolicies

table(Key,Data).evaluation:immediate ← ground(Key).


credential(_).ontology: URI.

abbrev(_).explanation: "this condition checks..."
Protune
Policy Filtering Example

allow(download('file1234.pdf')) ?

'file1234.pdf' is not public

Alice does not know what authenticated means

Only shared predicates

allow(download(Resource) ←
  public(Resource).

allow(download(Resource) ←
  authenticated(User),
  hasSubscription(User).

authenticated(User) ←
  credential(C),
  C.type:'id'.

authenticated(User) ←
  declaration([ user=User,
               password=P ]),
  passwd(User,P).

hasSubscription('Alice').

hasSubscription('John').

passwd('Alice','$1234ab3').

passwd('John', '8%&ca').
Deployed Application Scenarios
Combination of Policies and Trust/Reputation Algs.

Reputation-based

Policy-based

accessGranted(Res) ⇐
credential(X, VISA),
X.type : credit card, X.owner : B.

trust(A,B, download(file), 80–100) ⇐
credential(X, VISA),
X.type : credit card, X.owner : B.
allow(visaCard) ⇐
credential(member(Requester), bbb),
trust(self, Requester, buying, X), X > 0.8.
in(trust(X, Y, A, L), reputation pckg : eval trust())


[ Bonatti, Duma, Olmedilla, Shahmehri. An Integration of Reputation-based and Policy-based Trust Management. Submitted for Publication ]

P. A. Bonatti, D. Olmedilla
RuleML'06 Tutorial: Semantic Web Policies
Nov. 10th, 2006 | 137
Protune
Analysis of the Language (I)

- Well defined semantics
  - Yes
- Declarative
  - Yes
- Monotonicity (respect to policies, credentials and actions)
  - Yes
- Type of Evaluation
  - Distributed
- Use of Variables
  - Yes
- Operations/Combinations (conjunction, disjunction, negation, xor, etc.)
  - Conjunction, Disjunction, Negation
  - Extensible
- Management of Attribute Credentials
  - Yes
- Delegation of Authority
  - Yes
Protune
Analysis of the Language (& II)

- After-Disclosure Control
  - No
- External functions / execution of actions
  - Yes
- Ontology support
  - Yes
- Rule Support
  - Yes
- Protection of policies
  - Yes
- Extensibility
  - Yes
- Lightweight vs. Strong Evidence
  - Yes, explicit
- Usability
  - Logic language
Protune
Analysis of the Framework

- Conflict resolution / combination of policies
  - Does not apply
- Accountability / Proof carrying code
  - No
- Implementation
  - Ongoing
- Tools / applications
  - RCP Editor
  - Compatible with PeerTrust framework: integration into Web servers and Grid environments
- Support Explanations
  - Yes. Implemented
PeerAccess

Overview

Model and reason about distributed authorization in distributed systems

- Distributed reason on peers
- Control over disclosed information
- Hints specifying search space for answers

Composed of

- A modal language: base language
  - Specifies basic access control policies and related rules
- A modal meta-language
  - Determine the dynamic behavior of the system

[Winslett, Zhang, Bonatti. Peeraccess: a logic for distributed authorization. CCS 2005]
PeerAccess
Overview

Base policies

- A signs L ← ...
  - L is directly signed by A
  - A has digitally signed L and it was received by P

- A lsings L ← ...
  - L is logically signed by A
  - P has nonrepudiable evidence that A would sign L if shown such evidence

Release policies (sticky policies)

- A signs srelease (L,S,R) ← ...
  - A allows dissemination of L from S to R if L is true at S
  - Signer of a particular piece of information retains control over its future dissemination
PeerAccess
Example

Bob:

Bob \texttt{Isigns auth(shaketable,X)} \leftarrow

\texttt{CAS signs auth(shaketable,X)}

Bob \texttt{Isigns srelease(Bob signs auth(X,Y), Bob, Y)}
Bob \texttt{Isigns srelease(Bob signs auth(X,Y), Y, X)}
Bob \texttt{Isigns srelease(Bob signs auth(X,Y), Z, W)} \leftarrow

\texttt{Z != Bob,}

\texttt{Y Isigns condRelease(Bob signs auth(X,Y), Z, W)}

Alice:

Bob \texttt{signs auth(shaketable,Alice)}
Bob \texttt{signs srelease(Bob signs auth(X,Y),Y,X)}
PeerAccess
Analysis of the Language (I)

- Well defined semantics
  - Yes
- Declarative
  - Yes
- Monotonicity (respect to policies, credentials and actions)
  - There is no negation
- Type of Evaluation
  - Distributed
- Use of Variables
  - Yes
- Operations/Combinations (conjunction, disjunction, negation, xor, etc.)
  - Conjunction, Disjunction
- Management of Attribute Credentials
  - Yes
- Delegation of Authority
  - Yes
PeerAccess
Analysis of the Language (& II)

- After-Disclosure Control
  - Yes, in cooperative environments
- External functions / execution of actions
  - No
- Ontology support
  - No
- Rule Support
  - Yes
- Protection of policies
  - Yes, through disclosure policies
- Extensibility
  - Yes, via libraries
- Lightweight vs. Strong Evidence
  - Yes.
- Usability
  - Logic language
PeerAccess
Analysis of the Framework

- Conflict resolution / combination of policies
  - Does not apply

- Accountability / Proof carrying code
  - Yes

- Implementation
  - No

- Tools / applications
  - Not known

- Support Explanations
  - No
Other Policy Languages
Not covered in the tutorial

- PolicyMaker
- REFEREE
- Keynote
- Policy Description Language (PDL)
- Ponder
- Delegation Logic
- SD3
- TPL
- Cassandra
- WS-Policy
- E-P3P
Outline

- Introduction
- Where are we?
- **Deployed Application Scenarios**
  - Application Scenarios
    - World Wide Web
    - E-Mail
    - Semantic Web Services
    - Grid
  - Other Implemented Features
    - Distributed Loop Detection
    - Explanations
- What is still missing?
- Conclusions
Application Scenarios
Deployed Application Scenarios
Negotiating on the Web


P. A. Bonatti, D. Olmedilla
RuleML'06 Tutorial: Semantic Web Policies
Nov. 10th, 2006
Deployed Application Scenarios
P3P and Policy Enforcement with REI

Improvement of user side support

- More effective preference language: REI
  - More expressive than P3P
  - Well defined semantics
  - Also enables web privacy enforcement mechanisms

- Extensible trust model
  - Based on social recommendations
  - In addition to certificate only based trust

[ Kolari, Ding, Shashidhara, Joshi, Finin, Kagal. Enhancing Web Privacy Protection through Declarative Policies ]
Deployed Application Scenarios
Policy protecting e-mail

- Scalable, attribute-based access control policy
- E-mail messages as access requests from senders
  - Requesting write access to a mailbox
- Integration into SMTP protocol
- Relays on some sort of sender’s authentication

[ Kaushik, Ammann, Wijesekera, Winsborough, Ritchey. A Policy Driven Approach to Email Services ]
Deployed Application Scenarios
Policy Matchmaking for Semantic Web Services

- Proposed ontologies to model high-level security requirements and capabilities
- Policies are symmetric
  - They may constrain both client and service
- Extends OWL-S with REI policies
- Matching of client request with appropriate services
  - Using a Matchmaker, a capability-based matching engine
  - Verify compatibility of requester’s policies and the provider’s

Deployed Application Scenarios
Automatic Credential Fetching on Grids (I)

- Too many Credentials to keep track of
- Knowing which credential to use

MyProxy Credential Repository

0a Request previously stored proxy certificate
0b Receive proxy certificate

1 Mutual Authentication (M.A.)

Job must know in advance what credentials will have to be disclosed

Alice submitting a job

In large projects, an account per user does not scale

Authorization may depend on user’s properties
E.g. user’s affiliation with a project

- Different sites trust different CA
- No way to determine automatically which issuers are trusted


P. A. Bonatti, D. Olmedilla
RuleML'06 Tutorial: Semantic Web Policies

Nov. 10th, 2006 | 154
Deployed Application Scenarios
Automatic Credential Fetching on Grids (II)

Both client and servers are semantically annotated with policies

Annotations

■ specify constraints and capabilities
  ■ access control requirements
    ■ which certificates must be presented to gain access to it
    ■ who is responsible for obtaining and presenting these certificates

■ are used during a negotiation
  ■ to reason about and to communicate the need to see certain credentials from the other party
  ■ to determine whether requested credentials can be obtained and revealed.

User involvement is drastically reduced in favor of automated interactions.

[Constandache, Olmedilla, Siebenlist, Nejdl. Policy-driven negotiation for authorization in the semantic grid. 2005.]
Deployed Application Scenarios
Automatic Credential Fetching on Grids (& III)

- Distributed authorization mechanisms
  - Driven by policies, not hardcoded
- Bilateral policy specification
- Access is negotiated
- Dynamic credential fetching
  - Now possible to use discovery and scheduling services to locate the best available resources
  - Otherwise, impossible to predict before hand what exact service instance would be used and which certificates required
- Capability based authorization architecture
  - Instead of identity based
- No previous trust relationships required
- Monitoring and explanation of authorization decision
Other Implemented Features
Deployed Application Scenarios

Loop Detection: CIA Agents

I show you my CIA badge
If you show me yours first

[ Li, Du, Boneh Winsborough, Seamons, Jones. Oblivious Signature-Based Envelope
DARPA ACM Symposium on Principles of Distributed Computing, 2003]
Deployed Application Scenarios

Loop Detection: Distributed Tabling


P. A. Bonatti, D. Olmedilla

RuleML’06 Tutorial: Semantic Web Policies

Nov. 10th, 2006
Deployed Application Scenarios
Inference Web Answer Explanation

Provides generic explanation tools for (Semantic) Web based systems

- Infrastructure for presenting and managing explanations
  - Knowledge provenance
  - how answer were derived or retrieved

- IWBase
  - Web-registry with information sources, reasoners, languages, rewrite rules

- Proof Markup Language (PML)
  - Encoding of portable proofs

- IW Browser
  - Tool supporting navigation and presentation of proofs and their explanations

- No support for explaining infinitely failed derivations

Deployed application scenarios
Protune's explanations: Requirements - solutions

- Easy instantiation in any given application domain
  - One extra step: create literal verbalization rules
- Performance
  - Constructed at client side
- Explanation method
  - Focus on the aspects that are relevant to the user
  - Optional detailed view
  - Queries: why/why-not, how-to, what-if
- Presentation strategies
  - Simultaneous local + global information new!
  - Explanations are (potentially cyclic) hypertexts
- Explaining infinite failure
  - Tabled explanation structures new!

[Bonatti, Olmedilla, Peer. Advanced Policy Queries. REWERSE report I2-D4 and ECAI'06 ]
Why-Not Queries
Pruning strategies

I CAN’T PROVE THAT
it is allowed to download paper14.pdf

BECAUSE

Rule [r3] is not applicable:
THERE IS NO User SUCH THAT
User is authenticated

AND

Rule [r4] is not applicable:
THERE IS NO User SUCH THAT
User is authenticated

MOREOVER
THERE IS NO User SUCH THAT
User has paid for paper14.pdf

new!

FILTERED POLICY
[r3]: allow(download(Resource)) ←
authenticated(User),
blurred(hasSubscription(User)).

[r4]: allow(download(Resource) ←
authenticated(User),
paid(User,Resource).

METAPOLICY
allow(download(Resource)).explanation:
[it,is,allowed,to,download,Resource].

public(Resource).explanation:
[Resource,is,public].

authenticated(User).explanation:
[User,is,authenticated].

hasSubscription(User).explanation:
[User,has,subscription].

paid(User,Resource).explanation:
[User,has,paid,for,Resource].

[details]
**Why-Not Queries**

**Pruning strategies**

“authenticated” depends on a credential. “hasSubscription” depends on “authenticated”

---

**I CAN’T PROVE THAT**

it is allowed to download paper14.pdf

**BECAUSE**

**Rule [r3] is not applicable:**

THERE IS NO User SUCH THAT 
User is authenticated

Pruning: User is not authenticated so it makes no sense to inspect her subscriptions

**Rule [r4] is not applicable:**

THERE IS NO User SUCH THAT 
User is authenticated

MOREOVER 
THERE IS NO User SUCH THAT 
User has paid for paper14.pdf

**new!**

**FILTERED POLICY**

[r3]: allow(download(Resource)) ← 
authenticated(User), 
blurred(hasSubscription(User)).

[r4]: allow(download(Resource) ← 
authenticated(User), 
paid(User,Resource).

**METAPOLICY**

allow(download(Resource)).explanation:

[it,is,allowed,to,download,Resource].

public( Resource ).explanation: 
[Resource,is,public].

authenticated( User ).explanation: 
[User,is,authenticated].

hasSubscription( User ).explanation: 
[User,has,subscription].

paid( User,Resource ).explanation: 
[User,has,paid,for,Resource].
I CAN’T FIND ANY User SUCH THAT
User is authenticated
BECAUSE

c012 is a credential with
type ‘id’, name ‘John’ and issuer ‘L3S’
BUT
IT IS NOT THE CASE THAT
‘L3S’ is trusted for ‘id’

AND

Rule [r7] is not applicable:
THERE ARE NO User AND P SUCH THAT
username = User and password = P

POLICY
[r6]: authenticated(User) ←
  credential(Credential),
  Credential.type:’id’,
  Credential.name:User,
  Credential.issuer:CA,
  blurred(trusted_for(CA,’id’)).

[r7]: authenticated(User) ←
  declaration([ user=User,
               password=P ]),
  blurred(passwd(User,P)).

METAPOLICY
authenticated(User).explanation:
  [User,is,authenticated].

trusted_for(CA,Type).explanation:
  [CA,is,trusted,for,Type].

passwd(User,P).explanation:
  [P,is,the,correct,password,for,User].
Why-not demo
Sample screenshot

it is not allowed to download paper_0123.pdf because:

- Rule [3] cannot be applied:
  - paper_0123.pdf is not public [details]
- Rule [4] cannot be applied:
  - I find no User such that the User is authenticated [details]
- Rule [5] cannot be applied:
  - I find no User such that the User is authenticated [details]
  - I find no User such that the User paid for paper_0123.pdf [details]

Policy file
Why-not demo
Sample screenshot

it is not allowed to download paper_0123.pdf because:

- Rule [3] cannot be applied:
  - paper_0123.pdf is not public [details]
- Rule [4] cannot be applied:
  - I find no User such that the User is authenticated [details]
- Rule [5] cannot be applied:
  - I find no User such that the User is authenticated [details]
  - I find no User such that the User paid for paper_0123.pdf [details]

Policy file
Why-not demo
Sample screenshot

the User is not authenticated because:

- Rule [7] cannot be applied:
  - I find no Credential such that the Credential is an id [details]
- Rule [8] cannot be applied:
  - I find no Form such that the Form is a declaration [details]
- Rule [9] cannot be applied:
  - the procedure on http://lol.com/register.php has not (yet) been successfully completed [details]

Policy file
Why-not demo
After one more step...

the Card is not a valid credential because:

- Rule [19] cannot be applied:
  - c012 is a credential whose *issuer* is Open University

  **but**

  - I find no Key such that the Key is the public key of Open University

Policy file
Outline

- Introduction
- Where are we?
- Deployed Application Scenarios

**What is still missing?**
  - Independently of the SW
  - Open problems for SW researchers

- Conclusions
Widely recognized problems
A summary

- Integrating different rule types
  - for supporting multiple policy types

- Integrating strong, soft, lightweight evidence
  - therefore discrete + numeric trust models

- User awareness & control
  - high-quality explanations
  - controlled NL policies
Some problems we couldn't deal with not SW-specific

- Negation as failure and strong negation
- Mapping high-level policies onto low-level mechanisms
  - abstractions and approximations
- Validation & verification
- Policy composition
  - modules
- Hints for credential discovery
What's new in SW scenarios?

- Security/Privacy/Trust community addressed
  - Open systems
  - Heterogeneous software interoperability
  - Deployment on the web

- No new requirements regarding
  - Public/private nature of policies
  - Stateful/stateless nature of negotiations
  - Unilateral/bilateral forms of negotiations
Policies are still sensitive
⇒ not necessarily public

Business policies
- May reveal dishomogeneous treatment of different users
  - Which may irritate some customers
- May reveal strategic agreements with other companies

Private information
- Example: protecting family pictures
  - Only my friends can download these pictures
  - Some people may realize they are not friends by reading the policy
The Web supports transactions
⇒ negotiations can be stateful

Even if HTTP is stateless

- Many major web sites support transactions
  - Despite heavy traffic load
  - No convincing scalability issues
- Stateful protocols can be simulated
- Drawback of stateless approaches
  - Burden and responsibility on the programmer
  - Vulnerabilities (e.g. *cookies*)
Servers may release credentials
⇒ negotiations may be bilateral

Consider certifications and seal programs

- Publishing these credentials is good advertisement
  - Attracting potential customers
  - Making the service more competitive
- Not necessarily affecting negotiation length
  - Certifications are public
  - May be released all at once
  - On a public repository or on-demand if credentials are too many
  - The server may issue one hint to point to a repository
Within the realm of SW and KR&R and not in the focus of the trust community

- Ontology-based interoperability
  - including Pervasive lightweight evidence
- Regard policies as KBs
  - One knowledge – many uses
- Focus on intelligent interfaces
  - Explanations
  - Controlled NL front-ends
- Reasoning about policies
  - Select services based on their policies
  - Policy verification and validation
- Intelligent negotiation
  - e.g. Credential selection (cf. ASP tutorial)
Within the realm of SW and KR&R and not really tackled by security people

- Record linkage
  - Join data sources to infer sensitive information

- Inference problem
  - Possibly using common knowledge and user knowledge
  - Theoretical models exist (e.g. [Biskup et al.]), but

- Currently not checked by real systems
  - No machine-understandable model of available knowledge is implemented

- Ontologies and semantic markup
  - Enable automated inference-based attacks, but also
  - Enable automated inference checking
  - Using the same techniques (like password crackers)

[Biskup, Bonatti. DKE 01, FoIKS 02, ESORICS 02, IJIS 04, AMAI 04, FoIKS 2006]
Inference of sensitive information in the semantic web

Lots of information is implicit in published information

- Salaries can be inferred from roles
  - Salaries can be approximated from house value
- Phone number and zip codes are related
- ...

Common knowledge can be encoded

- In a machine understandable way
- Inference can be automated
  - Not tackled by current access control systems
Inference of sensitive information in the semantic web

Protecting semantic data

- Naturally subject to inference
- Opportunity for high-level specifications
  - Protect concepts (e.g. "my identity")
  - Use semantic techniques to identify data that encode sensitive concepts
  - Easier for untrained users
- It requires extensive and reliable tagging
  - Security people would not be convinced today
### Record linkage

The table below shows medical data released as anonymous. Each record includes sensitive data such as SSN, Name, Ethnicity (Ethn), Date of Birth (DOB), Sex, ZIP code (ZIP), and a problem such as Obesity.

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Ethn</th>
<th>DOB</th>
<th>Sex</th>
<th>ZIP</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>White</td>
<td>09.15.61</td>
<td>F</td>
<td>94142</td>
<td>Obesity</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Voter List

The voter list contains personal information such as Name, Address, City, ZIP code, Date of Birth, Sex, and Party. Sue Carlson is highlighted as a point of interest.

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>City</th>
<th>ZIP</th>
<th>DOB</th>
<th>Sex</th>
<th>Party</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sue Carlson</td>
<td>900 Market St.</td>
<td>San Fran.</td>
<td>94142</td>
<td>09.16.61</td>
<td>F</td>
<td>Democrat</td>
<td></td>
</tr>
</tbody>
</table>
Record linkage in the semantic web

Knowledge and reasoning facilitate

- Finding “linkable” data sources
- Joining heterogeneous data
  - Different attribute names
  - Different formats
- Using implicit information (inference)
Outline

- Introduction
- Where are we?
- Deployed Application Scenarios
- What is still missing?
- Conclusions
Conclusions

Policies are semantic markup

- Describing behavior (vs. content)
  - An instance of SW ideas
  - With widespread potential impact
  - In a short term
  - ...and in the long term (visionary perspectives)

- A case for rule-based ontologies
  - Novel interplay between the two towers

- Requirements for markup languages to be further analyzed
Conclusions

Plenty of possible SW contributions to security, privacy and trust & beyond

- Powerful KR&R infrastructure
  - Lightweight but expressive languages, and
  - Fast engines
- Knowledge-based policy handling
  - Enforcement, validation, explanations
- First concrete approaches to
  - Inference attacks
  - Preventing record linkage
Conclusions

Avoid pitfalls

- Wrong assumptions
  - Incompatible with realistic scenarios
  - ...recall conflicts...

- Re-inventing the wheel
  - There are already lots of high-quality works
  - Intersection between security/trust and KR&R communities
  - There are enough (really) new problems to be tackled!
Questions?
More exhaustive list can be found on the REWERSE deliverable page
- Deliverable I2-D8
References
Conferences (I)


6. Pranam Kolari, Li Ding, Shashidhara Ganjugunte, Anupam Joshi, Timothy W. Finin, and Lalana Kagal. Enhancing web privacy protection through declarative policies. POLICY 2005
References
Conferences (II)

1. Pranam Kolari, Li Ding, Shashidhara Ganjugunte, Anupam Joshi, Timothy W. Finin, and Lalana Kagal. Enhancing web privacy protection through declarative policies. POLICY 2005
6. Rita Gavriloaie, Wolfgang Nejdl, Daniel Olmedilla, Kent E. Seamons, and Marianne Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. ESWS 2004
References
Conferences (III)


References
Conferences (IV)


References
Conferences (& V)


References

Journals

References
Books


References

Other


