Negotiating Trust on the Semantic Web

Daniel Olmedilla
L3s Research Center / Hanover University

Reasoning Web Summer School
Msida, Malta, 28th July 2005
Outline

- Motivation
- Background
- Policy Languages
  - PeerTrust
  - Protune
- Policy Specification & Management
- Application Scenarios
  - WWW
  - Grid Computing
- Further Work
Motivation
- Background
- Policy Languages
  - PeerTrust
  - Protune
- Policy Specification & Management
- Application Scenarios
  - WWW
  - Grid Computing
- Further Work
Motivation Scenario
Buying in Internet

- Bob wants to access an electronic AI book at “E-Book Store” (a web site he found while surfing in Internet)

- Previously, E-Book requires Bob to register providing full name, age, complete address, telephone and e-mail

- Bob does not mind to give his full name and age but he does not like to provide his complete address, telephone and e-mail. However, he does not have any other option so he does it (although he does not provide his real address and telephone).

- E-Book sells that book. Therefore now it asks Bob to provide his credit card information. Bob would not mind to buy the book because it is not too expensive and he is really interested in reading it. However, he has never heard about E-Book so he decides to not buy it
Assumption: I already know you---you have a local account!
Access Control
Requirements

**Systems**
- Traditional distributed environments
  - Close environments: providers and requesters are known in advance
  - Server must trust the client: unidirectional (registration)
- WWW, P2P, GRID: dynamic networks
  - Nodes are usually not known in advance
  - Trust between strangers is needed
  - Bi-directional access control required

**Users**
- Do not want to register at any site (tedious task)
- Want control over what information they disclose and to set levels of privacy
  - E.g. My first name has not the same level than my credit card number
- Want assurance about what other nodes will do with their information
  - They want to know about the other party
Outline

- Motivation

- Background
  - Policy Languages
    - PeerTrust
    - Protune
  - Policy Specification & Management
  - Application Scenarios
    - WWW
    - Grid Computing
  - Further Work
Trust Negotiation (I)

Properties

- Trust is based on parties’ **properties**
- Every party can define access control policies to control outsiders’ access to their sensitive resources
- Establish trust **iteratively** and **bilaterally** by the disclosure of certificates and by requests for certificates

[ Winsborough, Seamons, Jones. *Automated Trust Negotiation.* DARPA Information Survivability Conference and Exposition, 2000]

[ Winslett. *An Introduction to Automated Trust Negotiation.* Workshop on Credential-Based Access Control, 2002 ]
Trust Negotiation (& II) Example

**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
Outline

- Motivation
- Background
  - Policy Languages
    - PeerTrust
    - Protune
  - Policy Specification & Management
  - Application Scenarios
    - WWW
    - Grid Computing
  - Further Work
Based on guarded distributed logic programs

- Some new syntax features: @, $, signedBy

Examples:

E-Learn:

freeEnroll(Course, Requester) $ Requester →

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

Alice:

  policeOfficer(‘Alice Smith’) @ ‘California State Police’ $ Requester →
  member(Requester) @ ‘Better Business Bureau’ @ Requester |
  signedBy [‘California State Police’].

The term **policy** refers to:

- **Security Policies**: pose constraints on the behavior of a system
- **Trust Management Policies**: typically used to collect user properties in open environments
- **Business Rules**: statements about how a business is done

In addition, associated to policies one needs to execute actions. Therefore also relevant:

- **Action Languages**: used in reactive policy specification to execute actions

Although many approaches have been described to address the above points, there is no common solution, integrating them all in a single framework.
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)*]
<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, known</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>
Outline

- Motivation
- Background
- Policy Languages
  - PeerTrust
  - Protune

Policy Specification & Management

- Application Scenarios
  - WWW
  - Grid Computing

- Further Work
Protecting Resources (I)
Is it just that easy?
Management of policies is difficult
- Many resources to protect
- Big administration effort

Correctness of our policies
- Real-world policies are complex
- Errors might provide inappropriate access
Ontologies for Specification & Management
Sharing Policies For Common Attributes

Outline

- Motivation
- Background
- Policy Languages
  - PeerTrust
  - Protune
- Policy Specification & Management

Application Scenarios
- WWW
- Grid Computing
- Further Work
Implementation
Trust Agent Architecture

TRUST AGENT

Credentials

Inference Engine

Policies
Ontologies
Metadata

Negotiation module

Interface

Credential verification

Strategy evaluator
Application Scenario
Negotiating on the Web

[ Gavriloae, Nejdl, Olmedilla, Seamons, Winslett.
No Registration Needed: How to Use Declarative Policies and Negotiation to Access Sensitive Resources on the Semantic Web. 1st European Semantic Web Symposium ]

Daniel Olmedilla
Reasoning Web
July 28th, 2005
Application Scenario (II)

Grid Limitations

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

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

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

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

In large projects, an account per user does not scale
Application Scenarios (III) Negotiating on the Grid

[Basney, Nejdl, Olmedilla, Welch, Winslett. Negotiating Trust on the Grid. 2nd Workshop on Semantics in P2P and Grid Computing at WWW’04]

Daniel Olmedilla
Reasoning Web
July 28th, 2005
Outline

- Motivation
- Background
- Policy Languages
  - PeerTrust
  - Protune
- Policy Specification & Management
- Application Scenarios
  - WWW
  - Grid Computing

Further Work
Trust Management
Reputation-based vs Policy-based

Reputation-based

Trust Factor

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

Policy-based

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

[Staab, Bhargava, Lilien, Rosenthal, Winslett, Sloman, Dillon, Chang, Eusuff, Nejdl, Olmedilla, Kashya

[Bonatti, Duma, Olmedilla, Shahmehri. An Integration of Reputation-based and Policy-based Trust Management. Submitted for Publication ]
Further Work
REWERS E12 Objectives

- Multiple inheritance & conflicting policies in ontology hierarchies
- Advanced Query Answering: how-to, what-if, why
- Negotiation Strategies
- Integrate event-condition-action (ECA) rules
- Natural language front-end to the policy domain
  - Natural Language Processing (NLP)
  - Automatic generation of natural language explanations from proofs and filtered policies
Thanks!

olmedilla@l3s.de - http://www.l3s.de/~olmedilla/