Negotiating Trust and Security on the Semantic Web

Daniel Olmedilla
L3S Research Center and Hanover University

Intelligent Information Systems (IISLAB), Linköpings Universitet
Linköping, Sweden, 10th Jun. 2005
Outline

- Motivation
- Trust Negotiation
- PeerTrust
- Policy Specification
- REWERSE WG I2
- Further Work
- References & Publications
Outline

- Motivation
  - Trust Negotiation
  - PeerTrust
  - Policy Specification
  - REWERSE WG I2
  - Further Work
  - References & Publications
Motivation: 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
Traditional Access Control for Decentralized Systems

- Assumption: I already know you---you have a local account!
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
- Trust Negotiation
  - PeerTrust
  - Policy Specification
  - REWERSE WG I2
- Further Work
- References & Publications
Trust Negotiation

- 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
**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
- Trust Negotiation
- PeerTrust
- Policy Specification
- REWERSE WG I2
- Further Work
- References & Publications
PeerTrust Language

- 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’].
PeerTrust Agent Architecture
Trust Agent Architecture Implementation
Network Diagram
Outline

- Motivation
- Trust Negotiation
- PeerTrust

Policy Specification

- REWERSE WG 12
- Further Work
- References & Publications
Protection of Credentials
Problems

- 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
Sharing Policies for Common Attributes (I)
Sharing Policies for Common Attributes (& II)

Similar to method inheritance in object oriented programming languages
Ease policy management
  - Reduce administration effort
  - Reduce number of errors

But what happen if
  - there are exceptions?
  - policies must be refined at lower levels?
Composing and Overriding Policies (I)

P1 = {Signed by authorized authority}
P2 = {Non-commercial}
P3 = {Category not F}
P4 = {4 years experience}
P5 = {3 years experience}

Illinois driver = {P1, P2, P3, P2, P3, P5}
Texas driver = {P1, P2, P4}
Composing and Overriding Policies (II)

- Mandatory policies cannot be overridden
  - They are enforced at lower levels

- Default policies can be overridden
  - In order to deal with exceptions

- In any case, policies can be refined with extra constraints at lower levels
Protégé Plug-in
Outline

- Motivation
- Trust Negotiation
- PeerTrust
- Policy Specification
- REWERSE WG I2
- Further Work
- References & Publications
Motivation (I)

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
Motivation (& II)

Although many approaches have been described to address the above points, there is no common solution, integrating them all in a single framework.
The Rule Language (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)
The Rule Language
Design Assumptions

- Provisional actions are orthogonal
  - The action associated to any ground atom \( A \) cannot change the truth value of any other ground provisional atom.

- Exchange of filtered set of policies between parties
  - in order to avoid combinatorial explosion of requests

- Negation is not applied neither to provisional predicates nor to any predicate occurring in a rule head
### Metapolicies (I)

<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>
Metapolicies (& II)

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

- logged(Msg,File).action:’echo’+Msg+’>’+File.

- credential(_).ontology:URI.

- abbrev(_).explanation:”this condition checks…”
Outline

- Motivation
- Trust Negotiation
- PeerTrust
- Policy Specification
- REWERSE WG I2

Further Work

References & Publications
Some Further Work (I)

- Policy Specification
  - Multiple inheritance & conflicting policies in ontology hierarchies
  - Disjunction on policy composition?

- REWERSE
  - Integrate event-condition-action (ECA) rules as
  - Natural language front-end to the policy domain
    - Natural Language Processing (NLP)
    - automatic generation of natural language explanations from proofs and filtered policies
  - Query Answering: how-to, what-if, why
  - Integration of numerical-based trust computation and trust policies

- Negotiating on the Grid
Outline

- Motivation
- Trust Negotiation
- PeerTrust
- Policy Specification
- REWERSE WG I2
- Further Work

References & Publications
References

- REWERSE WG I2
  Policy Language, enforcement, composition
  http://www.reverse.net/I2/

- PeerTrust project
  http://sourceforge.net/projects/peertrust/

- Security Agent in an Applet
  http://www.l3s.de/~olmedilla/projects/trust/applet/instructions.html
Publications

- **No Registration Needed: How to Use Declarative Policies and Negotiation to Access Sensitive Resources on the Semantic Web**
  European Semantic Web Symposium 2004

- **The Pudding of Trust**
  IEEE Intelligent Systems Journal, Vol. 19(5)

- **Driving and Monitoring Provisional Trust Negotiation with Metapolicies**
  IEEE 6th International Workshop on Policies for Distributed Systems and Networks (POLICY 2005)

- **Negotiating Trust on the Grid**
  2nd Workshop on Semantics in P2P and Grid Computing at the 13th International World Wide Web Conference

- **Ontology-Based Policy Specification and Management**
  European Semantic Web Conference 2005

- **Discovery and Contracting of Semantic Web Services**
  W3C Workshop on Frameworks for Semantic in Web Services
Thanks!

Questions?

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