Security for OWL-S

Grit Denker (SRI International)

Tim Finin, Lalana Kagal (UMBC)

Katia Sycara, Massimo Paolucci (CMU)

DAML PI Meeting

New York, May 25-26, 2004

Security Annotations for OWL-S

- Goal: annotation and matchmaking of "*security aspects*" of web services, including
  - Requirements and capabilities of a web service
    - *briefly mentioned, as this has been presented in earlier PI meetings*
  - Enforced policies for authorization, privacy and confidentiality
- Approach
  - Ontologies for high-level security mechanisms (e.g., "protocols used by service" or "credentials accepted by resource") and for cryptographic characteristics of service parameters (e.g., "encrypted/signed input/output parameter")
  - Rei policy language
  - Extensions of OWL-S Profile to indicate web service requirements, capabilities and enforced policies
  - Design and implementation of security matching algorithms

OWL-S Profile Security Extensions

- Additional object properties
  - **securityCapability** and **securityRequirement**
    - subPropertyOf profile:parameter
    - range SecurityMechanism
  - **policyEnforced**
    - subPropertyOf securityRequirement
    - range rei:Policy
- Note: similar properties have been defined for a class “Agent” to support client-server model of WS applications
- see www.csl.sri.com/~denker/owl-sec/ for ontologies and examples

Security Mechanism Ontology

SecurityMechanism class

- with subclasses: Syntax, KeyFormat, Protocols, Signature, Encryption, SecurityNotation
- and object properties: relSecurityNotation, reqCredential, syntax, etc. [with appropriate range classes]
- imports: Credential ontology
  - Simple/Composed Credential
  - Certificates (X509, etc.), Keys, Login, Cookie, BioMetric, IDCard, etc

Why is this not enough ?

- Authorization only based on
  - Protocols supported
  - Credentials (login/password, certificate) required
- Need more expressive policies
  - Based on attributes of requester, service and other context
- Did not handle privacy at all
- Should be able to handle prohibitions as well
  - E.g.. No undergraduate student should be able to access this service

→ Policy-Based Security Infrastructure

Example policies

- Authorization
  - Policy 1: Stock service is not accessible after the market closes
  - Policy 2: Only members of the LAIT lab who are Ph.D. students can use the LAIT lab laser printer
- Privacy/Confidentiality
  - Policy 3: Do not disclose my my SSN
  - Policy 4: Do not disclose my telephone number
  - Policy 5: Do not use a service that doesn’t encrypt all input/output
  - Policy 6: Use only those services that required an SSN if it is encrypted

Specification of Policies

- Use of Rei policy specification language
- Authorization, Privacy and Confidentiality Policy are subclasses of Rei's Policy class
    - Authorization policies usually associated with services
    - Privacy & confidentiality policies usually associated with clients
- Authorization policies
    - Permissions & prohibitions over attributes of the requester, service, and the invocation context
- Privacy policies
    - Here: Restricting access to services satisfying I/O conditions
- Confidentiality policies
    - Here: Restrictions on cryptographic characteristics of I/O parameter
    - => Ontology for cryptographic characteristics of service parameters

Ontology: Cryptographic Characteristics of Parameters

- Classes **InfObject** (information object)
- Subclasses **EncInfObj** (encrypted inf. obj.) **SigInfObj** (signed inf. obj.)
- Object property of InfObj is **baseObject**
  - Describing the type or structure of the information that is encoded
- Further object property of InfObj is **cryptoAlgUsed**
  - Defining the algorithm used to encode the information
- Web service input/output parameters can be described as information objects that reference the type of information (e.g., SSN) and the kind of security technique applied to it (e.g., encryption or signature)
- Confidentiality policies use same approach

Rei Policy Language

- A declarative policy language for describing policies over actions
- Represented in OWL + logic-like variables
- Based on deontic concepts
  - Right, Prohibition, Obligation and Dispensation
- Conflict resolution through the use of meta policy specifications

Rei Example

- All members of the LAIT lab have the right to use action ‘printing’
- Constraint

```
<constraint:SimpleConstraint rdf:about="&labpolicy;members_of_lait"
    constraint:subject="&labpolicy;var1"
    constraint:predicate="&univ;affiliation"
    constraint:object="&labpolicy;LaitLab"/>
```

Unify

- Right

```
<deontic:Right rdf:about="&labpolicy;right_to_print”>
    <deontic:actor rdf:resource="&labpolicy;var1"/>
    <deontic:action rdf:resource="&labpolicy;printing"/>
    <deontic:constraint rdf:resource="&labpolicy; members_of_lait "/>
</deontic:Right>
```

Example

- Mary is looking for a reservation service
  - foaf description for Mary’s personal information
  - Confidentiality policy
    - Don’t use services that use unencrypted personal information, i.e., require input parameter of services to use encrypted personal information
  - Privacy policy
    - SSN should never be disclosed, i.e., forbid services that have as output an instance of type SSN
- BravoAir is a reservation service
  - OWL-S description
  - Authorization policy
    - Only users belonging to the same project as John can access the service

Mary’s FOAF Description

```
<!-- Mary's FOAF description -->

<foaf:Person rdf:ID="mary">

<foaf:name>Mary Smith</foaf:name>

    <foaf:title>Ms</foaf:title>

    <foaf:firstName>Mary</foaf:firstName>

    <foaf:surname>Smith</foaf:surname>

    <foaf:homepage
    rdf:resource="http://www.somewebsite.com/marysmith.html"/>

    <foaf:currentProject rdf:resource=" http://www.somewebsite.com/SWS-
    Project.rdf "/>

    <sws:policyEnforced rdf:resource="&mary;ConfidentalityPolicy"/>

</foaf:Person>

</rdf:RDF>
```

Bravo Authorization Policy

```
<entity:Variable rdf:about="&bravo-policy;var1"/>
<entity:Variable rdf:about="&bravo-policy;var2"/>
<constraint:SimpleConstraint
            rdf:about="&bravo-policy;GetJohnProject"
     constraint:subject="&john;John"
     constraint:predicate="&foaf;currentProject"
     constraint:object="&bravo-policy;var2"/>
<constraint:SimpleConstraint
             rdf:about="&bravo-policy;SameProjectAsJohn"
     constraint:subject="&bravo-policy;var1"
     constraint:predicate="&foaf;currentProject"
     constraint:object="&bravo-policy;var2"/>
<!-- constraints combined -->
<constraint:And rdf:about="&bravo-policy;AndCondition1"
     constraint:first="&bravo-policy;GetJohnProject"
     constraint:second="&bravo-policy;SameProjectAsJohn"/>
```

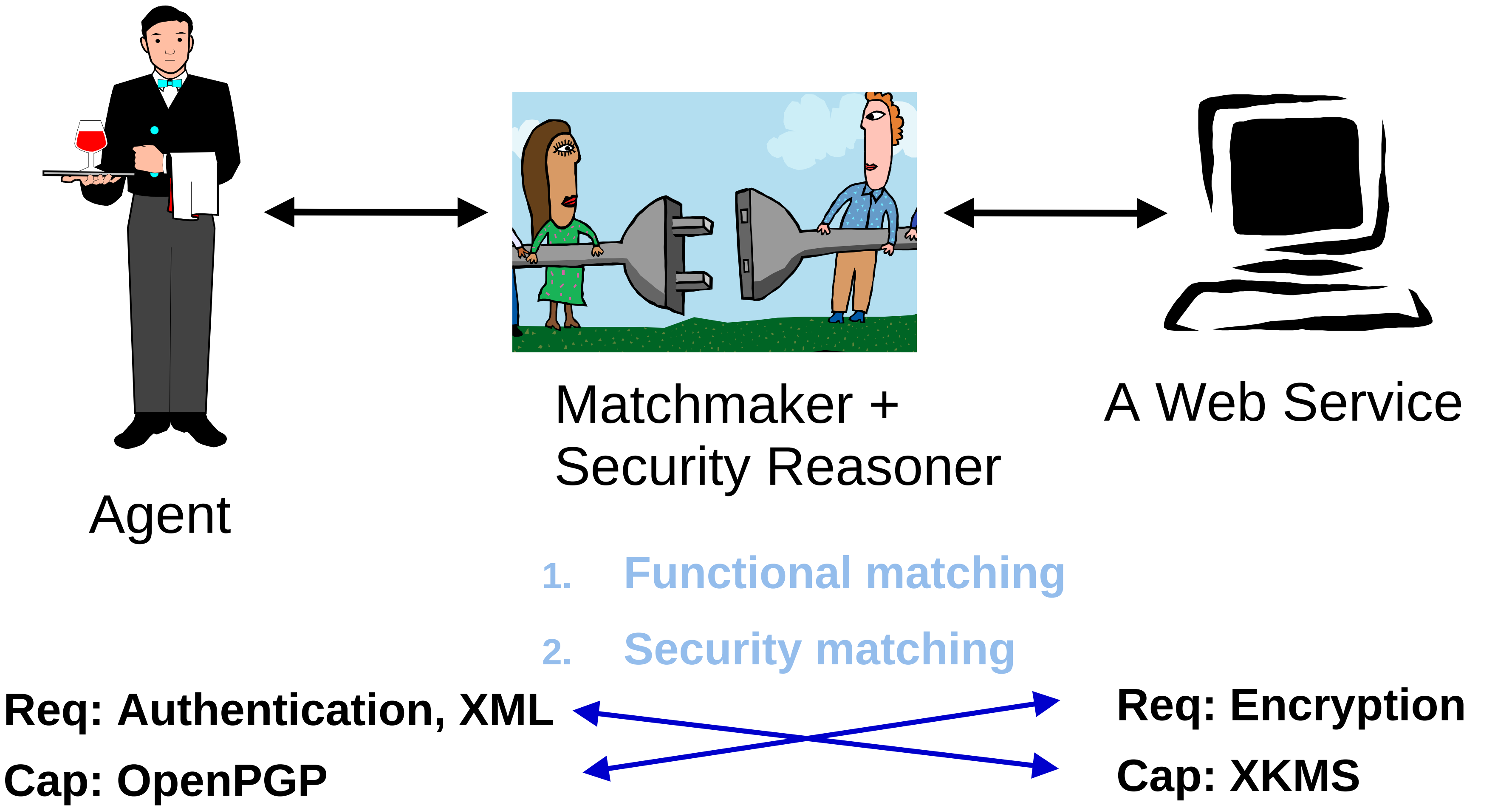
```
<deontic:Right rdf:about="&bravo-policy;AccessRight">
    <deontic:actor rdf:resource="&bravo-policy;var1"/>
    <deontic:action rdf:resource="&bravo-
    service;BravoAir_ReservationAgent"/>
    <deontic:constraint rdf:resource="&bravo-
    policy;AndCondition1"/>
</deontic:Right>
………
<rdf:Description rdf:about="&bravo-
    service;BravoAir_ReservationAgent">
    <sws:policyEnforced rdf:resource="&bravo-
    policy;AuthPolicy"/>
</rdf:Description>
```

Matching & Compliance Checking

- Matching of web service and agent security requirements and capabilities
  - Prototype implementation uses JTP
  - Integrated with CMU Matchmaker

- Compliance checking of policies
  - Design and implementation of algorithm for matching policies
  - Integration of the algorithm into CMU’s Matchmaker and OWL-S Virtual Machine (future work)

Matching Security Annotations



Agent

Matchmaker +
Security Reasoner

A Web Service

1. Functional matching
2. Security matching

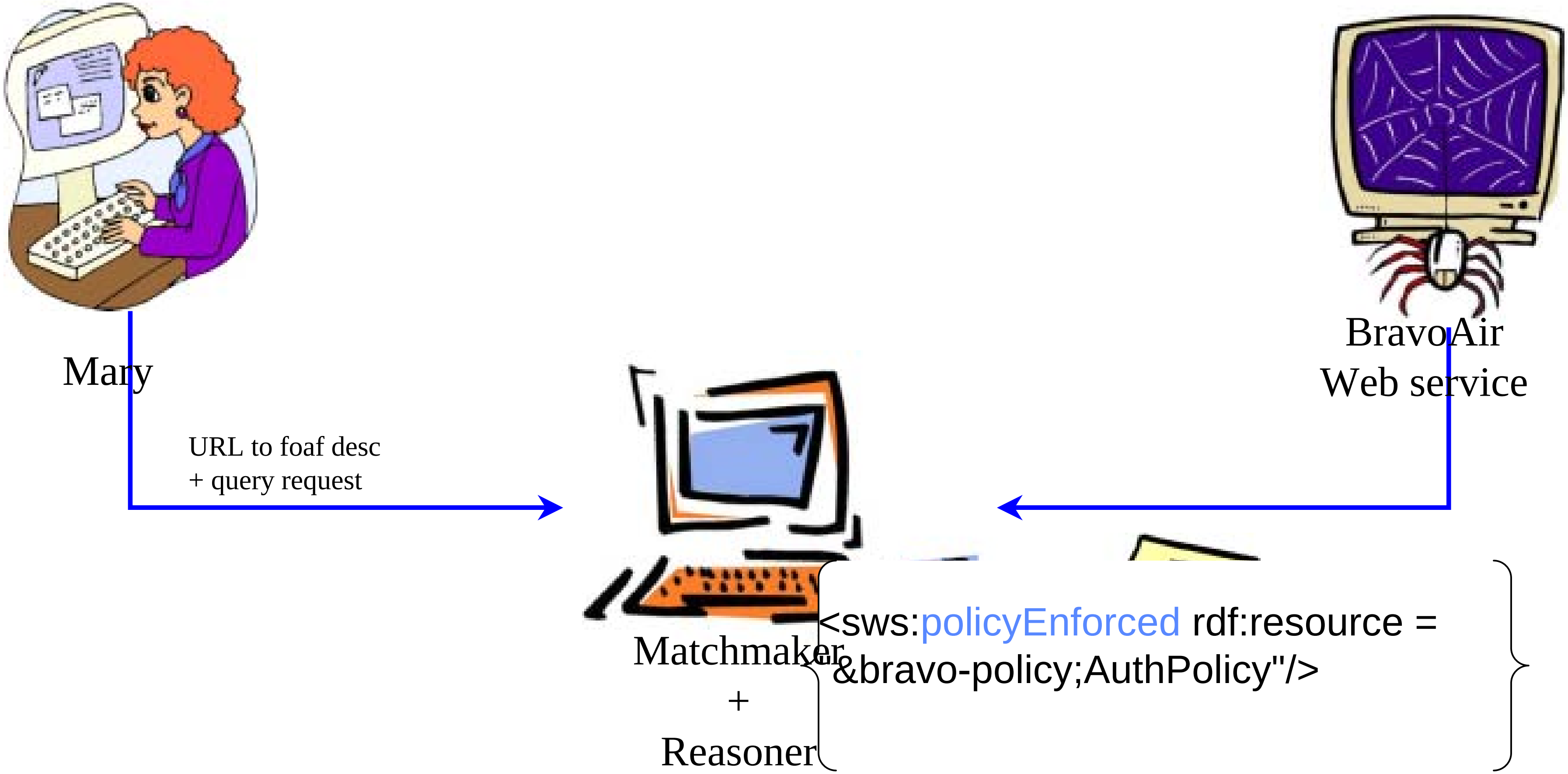
Req: Authentication, XML

Cap: OpenPGP

Req: Encryption

Cap: XKMS

Policy Compliance Checking



Mary

URL to foaf desc
+ query request

BravoAir
Web service

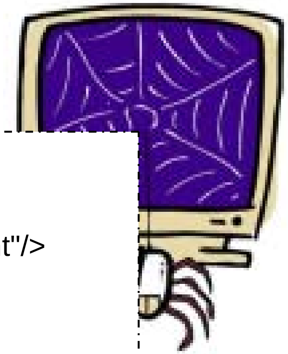
Matchmaker
+
Reasoner

<sws:policyEnforced rdf:resource =
"&bravo-policy;AuthPolicy"/>

Policy Compliance Checking

Mary

Mary’s query = Bravo Service ? YES
Extract Bravo’s policy



voAir
service

```
<deontic:Right rdf:about="&bravo-policy;AccessRight">
        <deontic:actor rdf:resource="&bravo-policy;var1"/>
        <deontic:action rdf:resource="&bravo-service;BravoAir_ReservationAgent"/>
        <deontic:constraint rdf:resource="&bravo-policy;AndCondition1"/>
</deontic:Right>

<policy:Granting rdf:about="&bravo-policy;AuthGranting">
        <policy:to rdf:resource="&bravo-policy;var1"/>
        <policy:deontic rdf:resource="&bravo-policy;AccessRight"/>
</policy:Granting>

<sws:AuthorizationPolicy rdf:about="&bravo-policy;AuthPolicy">
        <policy:grants rdf:resource="&bravo-policy;AuthGranting"/>
</sws:AuthorizationPolicy>

<rdf:Description rdf:about="&bravo-service;BravoAir_ReservationAgent">
        <sws:policyEnforced rdf:resource="&bravo-policy;AuthPolicy"/>
</rdf:Description>
```

var1 = http://www.cs.umbc.edu/~lkagal1/rei/examples/sws-sec/MaryProfile.rdf

Algorithm for Matching Policies

1. After the client sends a query request, MatchMaker finds a matching service and fetches its OWL-S description
2. It extracts the service's authorization policy from the policyEnforced attribute and sends it to the Rei Reasoning Engine along with the client's description
   - Rei returns true or false based on whether the client meets the authorization policy of the service. If false, matching failed.
3. The matchmaker extracts the client's privacy and confidentiality policies and sends it to the Rei Reasoning Engine along with the service's OWL-S description
   - Rei returns true or false based on whether the privacy and confidentialiy policies are met or violated. If false, matching failed.
4. Matching between client and service is complete

Some Open Questions

- Applicability of other policy languages
- Integration with WS* standards
- Enforcement of privacy, confidentiality and data integrity policies during execution
  - Confidentiality
    - One possible approach is for the OWL-S virtual machine to handle encryption/signing on behalf of the web service and the requester
  - Privacy
    - Reputation
    - Trusted third parties

Summary

- Contribution
  - Specification of security policies for web services
  - Authorization policies are enforced during discovery
  - Privacy and confidentiality policies are matched

Other Security-related Work

- Design and annotation of semantic security services
  - Grit Denker, Andrew Ton, Son Nguyen (SRI)
  - See http://www.csl.sri.com/~denker/owl-sec/SecurityServices/
- OWL-S Specification of Service Interaction Protocol
  - Grit Denker (SRI), Terry Payne and Ron Ashri (Univ. of Southampton, UK), Mike Surridge and Darren Marvin (IT Innovation, UK)
  - UK project “Semantic Firewall”
  - See http://www.csl.sri.com/~denker/owl-sec/sfw