Context-Aware Security Policy for the Service Discovery

Slim Trabelsi
Institut Eurecom
Slim.trabelsi@eurecom.fr

Laurent Gomez
SAP Labs France
laurent.gomez@sap.com

Yves Roudier
Institut Eurecom
yves.roudier@eurecom.fr

Abstract

Dynamic and self organizing systems like those found in pervasive computing systems or semantic web based scenarios raise numerous challenges regarding security and privacy. Service discovery is a basic feature of SOA deployment in such systems, given that entities need to locate services they can describe without an a priory knowledge. After inherent threats to service discovery in ubiquitous networks, we propose a registry based solution in which context-aware security policies are enforced in order to ensure privacy and access control for clients and services. We offer the possibility for the users to specify their security preferences that will be enforced during the discovery process. Experimental results based on an implementation of our approach are finally presented.

1. Introduction

Orchestration is becoming an essential feature for developing software for increasingly pervasive systems, in particular with the fast development of ubiquitous computing. The orchestration technique obviously comes at a cost: being able to locate previously unknown services becomes mandatory. The first orchestration technique applied generally is the service discovery that allows a dynamic detection of the available services in the network.

With the emergence of the Web Service technology, the discovery process should address the heterogeneity of services and platforms from a technical perspective, the complex semantics of service descriptions (e.g. resorting to terminology- or ontology-based descriptions), specific security and trust requirements, altogether with scalability. Web Service discovery solutions like UDDI [1], WS-Discovery [2], were developed to answer some of these requirements, yet still do not address most security and trust issues. In the WS-Discovery protocol for instance, security is limited to the use of signatures for verifying the integrity of messages. It is not sufficient to protect sensitive information about services from becoming available to rogue users; private information of a user might also get revealed to a service without any assessment of that service's potential maliciousness. This paper discusses how WS-Discovery may be extended to incorporate appropriate confidentiality and privacy protections restricting the potential matching between a client lookup request and a service profile. In particular, this paper describes how XACML may be used to implement such functionalities and how it needs to be extended to incorporate an evaluation of the context of a user or of the device hosting a service, an essential feature for enabling service discovery in ubiquitous computing.

This paper is organized as follows. In the section 2 we introduce the notion of service discovery and we dress a threat model related to this mechanism. In the section 3 we describe the proposed solution to overcome these security failures. In the section 4 we detail the architectural and the technical aspects of the implementation. We also provide the performances results obtained with our implementation. Finally we compare our approach with related work.

2. Service Discovery and Security

This section introduces service discovery concepts and goes on to discuss the threats attached to this mechanism.

2.1. Service Discovery Definition

Communication devices in fixed networks like local LANs traditionally are assigned a static network configuration, or at worst use DHCP to dynamically configure their IP address. With the emergence of new dynamic networks and services where devices are pervasive, the discovery techniques are being adapted in order to find mobile services rather than devices. This adaptation in particular addresses how to combine services as a logical layer in such systems together with the specification of environmental constraints.

Centralized discovery approaches rely on a registry which plays the role of yellow pages, and which clients can refer to. A service advertises its capabilities to the registry, which will store them for a certain amount of time. A client solicits the registry to find a service by sending a request containing service preferences, which the registry tries to match with the most suitable provider found from the stored advertisements. In that approach, registries have to be considered by the services and the clients as a third trusted party.

2.2. Revisiting Service Discovery Threats

This section discusses the threat model of service discovery services and in particular which parts of such systems would be worthy targets to adversaries.

2.2.1. Threats and Attacks

The main players of the discovery phase are the service requester (client) and the service provider (server), even in the case of a registry based service discovery. We try here to provide a global idea about threats and possible attacks that can be built against the data and resources of service discovery players.

Protocol Messages and Entities

- The registry is not available (service-side)
- Client request disclosure (client-side)
- Interception of request (client-side)
- Message modification or drop (client side)
- Replay of lookup message DoS (client-side)
- Replay of registration message (registry-side)

Service Registration

- Registration to a malicious registry (server-side): an attacker might fake being a registry
- A service can be deregistered by an unauthorized party (service-side):
- Wrong registration (registry-side): An attacker can send a fake registration message to the registry.

Matching Process

- Client lookup disclosure (client-side): client intentions or activity might be disclosed if the matching process is open to all services registered.
- Service discovered by unauthorized party (service-side)

This paper especially focuses on the development of appropriate policy specifications for the latter category of threats, and based on XACML. The policy enforcement mechanism described here makes it possible, regarding the last threat for instance, to specify authorized clients through the specification of their context or their devices physical situation, as acquired from the environment, as additional and dynamic attributes of the client in that example.

3. Context-Aware Access Control for Secure Discovery

This section introduces the architecture of our solution to supporting contextual attributes as supplementary constraints for matching client or service profiles at discovery.

3.1. Discovery Policy

The threat model exposed in the previous section makes it clear that clients should be able to find a service matching their preferences, both in terms of the characteristics of the service and in terms of security and privacy requirements imposed respectively by the service and by the client. On the client side, the user should be sure that only services matching his preferences would be returned: from his point of view, trusting a service should therefore go beyond the simple authentication of the service provider and also encompass a complete certification process of the capabilities of the service. On the server side, the problem is quite similar since the server does not know the users that can potentially gain access to its service. They should therefore be accessible only to client they trust to access them according to a precise behavior guaranteed by some authority.

Assigning the responsibility to enforce such discovery policies to a trusted entity of the system is therefore critical to service discovery. To avoid raising the complexity of service discovery, we do not propose to add a new entity to the system together with a dedicated protocol, but rather to assign this task to the registry. The choice of the registry as being the trusted third party in charge of the policy enforcement is an absolute requirement in centralized approaches, since matching already implicitly is a trusted operation, and matching and policy enforcement are closely tied together.

Discovery policies [3] may be quite simple: the client or the service provides rules that describe who can access their respective profile based on some attributes. In this paper the discovery policy objective is twofold:

- Access Control: discovery constitutes a preliminary form of access control to services by restricting the clients which will be able to subsequently contact a service. The sensitive resource here is the service's profile that must be hidden to the non authorized users.

- Privacy Protection: the client can protect the private information he reveals for each lookup he performs (identity, intentions, favorite services …) from an uncontrolled disclosure.

As shown in Figure 1 the usual discovery messages (publish and lookup) should be accompanied by some credential (certificate or key) in order to be authenticated by the registry, by a discovery policy that will be enforced by the registry in order to protect the entities according to their desires, the whole being secured using a signature based on the credential transmitted for instance.

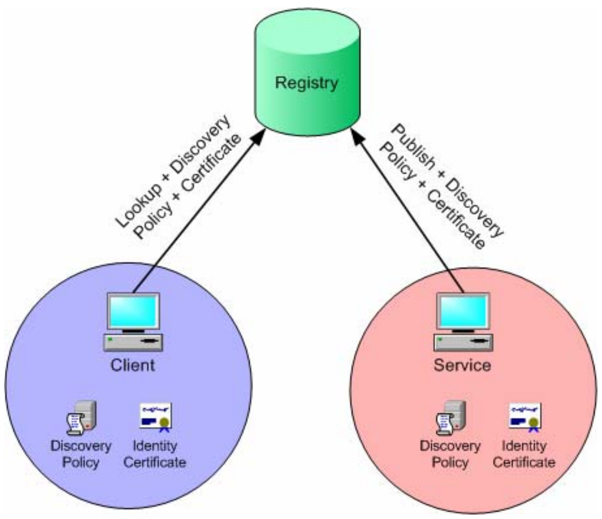

Figure 1: Communicating discovery policies

3.2. Context-Awareness

The use of context represents a significant benefit for enabling service discovery in the highly dynamic environments addressed in ubiquitous computing. Context or context information refers to any information that can be used to characterize the state of an entity (user, or software, or hardware component of a computing system) [4]. The location of a service, obtained for instance through a GPS or WiFi-based location of the device on which a service is running, network bandwidth, or the security protocols enabled on some platform all may serve to characterize dynamic services and networks. Service discovery may obviously exploit context to achieve more precise matching in such environments [5],[6]. More importantly, such context information complements and provides more flexibility to the discovery policy specification. It in particular makes it possible to express fine grained discovery policies more closely following the constant changes of the environment and services.

Approaches to the introduction of context-awareness for service discovery so far only exploited raw context directly acquired from sensors (e.g. GPS location, remaining battery). While this may indeed enhance service discovery with basic context-awareness, the use of sensor context information is however too restrictive for defining a discovery policy. We instead introduce semantically-rich context information, thereby supporting context reasoning: raw contextual data that are gathered from sensors, like the location, can therefore be further processed to derive complex information, such as the proximity. De facto, we already improve the flexibility of context-aware discovery policies whose expressive power extends to more complex contexts. Context reasoning also may take place during the enforcement of discovery policies, and makes it easier to combine context information coming from different sources.

3.3. Secure Context Acquisition

The use of context information for enhancing security mechanisms also make it necessary and even critical to assure the security of the context acquired [7]. While the proof of concept implementation presented in the next section does not address these issues, this section provides a non-exhaustive listing of various approaches that may permit to secure context acquisition.

3.3.1. Confidential Context Information

Considering user context information, one should be able to protect his personal information such as his health status or medical history (context information privacy), which touches the privacy of a user.

Several approaches target this issue [8] [9] [10] of user's privacy protection. They all aims at providing security control for controlling the disclosure of context information by the user.

3.3.2. Context Information Integrity

Integrity protection aims at guarantee that the context information acquired has not been corrupted by an unauthorized third party while in transit. Hashing and public key digital signature are two alternatives. But the latter, relying on public key infrastructure, may impose important constraints on highly decentralized and pervasive low-cost sensor networks.

4. Implementation design

Our prototype implementation of the system described above relies on three fundamental components that enable a secure and context-aware service discovery: a service discovery protocol that defines a standard for the message format, the exchange techniques and the registration management. A security policy module is used to generate, to reason about, and to enforce the policy used to secure the discovery process. A context-aware module is used to reason about the contextual information and data used during the discovery policy enforcement.

4.1. Service Discovery Protocol

We selected to extend an existing service discovery protocol called Web Services Dynamic Discovery (WS-Discovery). This protocol defines a multicast discovery scheme to locate services connected to a network (mostly assumed to be a LAN or WLAN). Each service provider announces itself (by sending a "Hello" message) through the multicast group to expose the services that can provide. Each user that is looking for a service propagates its query (by sending a "Probe" message) through the multicast and only the concerned service must make a response (by sending a "Probe Match" message). As we mentioned previously the default matched attributes are the Type and the

Scope of the service, obviously other attributes and meta-data information can also be.

The WS-Discovery specification does not suggests securing the discovery process but it recommends the usage of a compact signature format to secure the exchanged messages. In this case each entity has the possibility to verify the signature of the message sender. This signature protects against the message modifications the replay, the spoofing.

Signature verification is obviously insufficient to protect users (servers and clients) since a valid signature only assess that the message content has not been altered without presuming of the level of trust of the issuer. Moreover, the content of the message is not confidential and there is no guarantee against the disclosure of private information. For example a malicious server can publish fake services with a valid signature or listen to request messages in order to collect valuable information.

4.2. Context Reasoning Module

In order to ease the definition of discovery policies exploiting context, context handling and reasoning relies on the use of a context ontology. This section describes this ontology and how it can be combined with service discovery.

4.2.1. Context Information Representation

Ontologies aim at classifying, characterizing and at last at establishing relationship between concept in a given domain. Therefore, they provide a strong support for reasoning about concept. Regarding context ontology, we decide to use the Context OntoLogy (CoOL[11]). It is expressed with the Web Ontology Language (OWL[12]). For sake of efficiency, we use the OWL-DL version of OWL.

4.2.2. Reasoning about Context Information

For reasoning about context information, we distinguish two complementary approaches for reasoning about context information: ontology and inference rule based reasoning. As described in the previous section, ontology supports relationship definition between context information. Based on those relationships, ontology eases reasoning about context. In Figure 2, we express in OWL-DL the following reasoning: “if patient pulse is below 10 and his body temperature is below 10, then he is unconscious”. We establish a relationship between patient’s pulse and body temperature in order to infer on his health condition.

```
<xsd:simpleType name=below10>
<xsd:restriction base=xsd:positiveInteger>
<xsd:maxInclusive value=10>
</xsd:restriction>
</xsd:simpleType>
<owl:Class rdf:ID=isUnconscious>
  <owl:intersectionOf rdf:parseType=Collection>
    <owl:Class rdf:about=#User/>
    <owl:Restriction>
      <owl:onProperty rdf:resource=#Pulse>
      <owl:someValuesFrom rdf:resource=#below10>
    </owl:Restriction>
  </owl:intersectionOf>
  <owl:intersectionOf rdf:parseType=Collection>
    <owl:Class rdf:about=#User/>
    <owl:Restriction>
      <owl:onProperty rdf:resource=#BodyTemperature>
      <owl:someValuesFrom rdf:resource=#below10>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

Figure 2: OWL-based reasoning sample

Moreover, ontology enables us to express similarity relationship between context. Thus, we can express a similiraty between pulse and heart rate. If heart rate is not available, then pulse can be used in order to reason about patient’s health condition.

Nevertheless the expressiveness power of OWL can be quickly restrictive as soon as we try to target more complex reasoning about context. Due to the restriction of OWL-DL, ontology based reasoning is limited to binary relationship between two context notions. For this reason, we can not quantify relationship in OWL-DL. For example, proximity relationship can be established in context ontology, but it can be quantify with respect to the distance between the users.

For those reasons, we propose to use inference rule-based reasoning engine such as Jess [13], in combination to ontology-based reasoning. Once ontology-based identified relationships between context, inference rule tends to cope with ontology-based reasoning by evaluating and quantifying those relationships. In our use case, proximity between doctor and patient has to be evaluated. In Figure 3, we provide an example of inference rule about proximity between a physician and a patient. This rule is defined for Jess. We intentionally skip the acquisition of physician and patient’s location in the rule definition.

```
;; ACQUIRE PHYSICIAN AND PATIENT LOCATION
;; TEST IF THEY ARE NO FAR THAN 2000 METERS
=>
(assert
   (triple
             (p "isCloseTo")
             (s ?PatientLocation)
             (o ?PhysicianLocation) )  )
```

Figure 3: Inference rule sample with Jess [13]

4.3. Security Policy Module

In [17], we propose a framework for exploiting reasoning about context at access control policy enforcement based on XACML. XACML request consists of a triple {Subject, Resource, Action}. A Subject tends to gain access to a Resource (e.g. file, web service) in order to perform an Action (e.g. read/write, invoke a method). The Subject is characterized by a set of attributes (e.g role, location). Based on this triple {Subject, Resource, Action}, a rule-based access control policy is enforced. After decision making, a XACML response is sent back to the requestor (e.g. Permit, Deny, Intermediate or Not Applicable).

Using the XACML policy language, we can easily restrict the discovery to some authorized clients as illustrated in Figure 4. In this example, we restrict access to getPatientMedical action of any resource to a user characterized with a physician role. (For sake of clarity, we skip namespaces.)

```
   <!--  Check if the subject is physician   -->
   <Condition FunctionId="function:string-equal">
      <Apply FunctionId="function:string-one-and-only">
         <SubjectAttributeDesignator DataType="string"
AttributeId="SubjectRole" />
      </Apply>
      <AttributeValue
DataType="string">Physician</AttributeValue>
      </Condition>
   </Rule>
</Policy>
```

Figure 4: XACML Policy Definition

We can extend this security policy definition with the following condition.

```
<Apply FunctionId="CloseTo">
  <Apply FunctionId="findLocation">
    <SubjectAttributeDesignator
      DataType=GPSLocation AttributeId="SubjectLocation"/>
  </Apply>
  <AttributeValue
     DataType="integer">2000 meters</AttributeValue>
  <Apply FunctionId="string-one-and-only">
    <SubjectAttributeDesignator
      DataType=string AttributeId="PatientID" />
  </Apply>
</Apply>
```

Figure 5: CloseTo Conditon in XACML

Figure 5 describes how to infer the proximity between patient and physician. If the distance between their two personal devices is below 2000 meters, than we consider them as close to each other.

4.4. Architecture

In this section we detail how our implementation is working and how the different entities represented in the Figure 6 can interact with each others. Step (1) is initiated by the server in order to register its services by sending a Hello message containing the description of its capabilities, its profile (Also credentials) and some specific contextual information. The step (2) is the Client's service lookup by sending a Probe message containing the service request and a credential (to be authenticated). The step (3) performed by the registry consists in a request matching with the existing service profiles. If needed the client (and services are authenticated in (4), then a reasoning about the eventual contextual information is performed in (5), Policy enforcement is done in the step (6). If the request is accepted, the registry returns a response to the client by sending a ProbeMatch message (7).

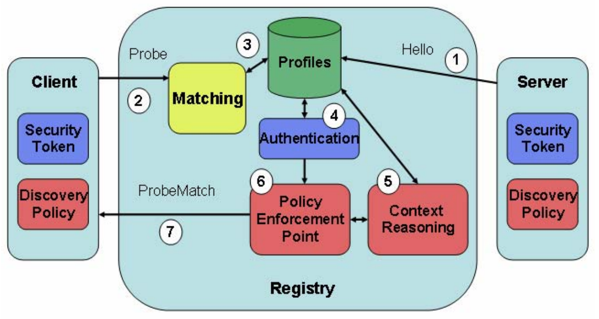

Figure 6: Global Architecture Design

4.5. Performance and results

In order to evaluate the efficiency of our solution we developed a Java prototype and we performed some measurements about time execution and memory consumption. For these experiments we used:

- OS: Fedora Core 5 with a Linux 2.6.x kernel i686
- CPU: Mobile Intel Pentium 4 CPU 1.70 GH
- Physical Memory 512 MB

In this table we provide all the measurement values related to each step described in section 4.4.

	Actions	Time (ms)	Size (byte)
(1)	Sending Hello	31	3963
(2)	Sending Probe	67	862
(3)	Service matching	370	-
(4)	Authentication	1572	-
(5)	Context Reasoning	4005	76000
(6)	Policy enforcement	862	-
(7)	Sending ProbeMatch	15	1622

5. Related Work

To our knowledge, the notion of context-aware security policy for the service discovery has not been investigated in the literature, while context aware service discovery or secure discovery is not new any more. This section gives an overview of some interesting work regarding these two topics.

One of the first approaches dealing with secure service discovery was proposed by [14]. This architecture relies on an additional component, called Service Discovery Service (SDS), which plays the role of a secure information repository (secure registry). It provides authentication, access control, encryption, signature verification, and privacy protection using a PKI. Contrary to our solution, clients and services do not have any possibility to define their own security preferences regarding discovery.

In a precedent work [15] we proposed a specific solution to secure peer-to-peer service discovery mechanisms. This solution is based on the use of an Identity Based Encryption scheme to protect the requests and the announcement messages. This solution needs not relying on a trusted third party in order to perform a secure service matchmaking.

[16] addresses privacy protection aspects of the discovery process. The authors propose the use of Bloom filters to protect the client and server personal information. Membership tests are performed between the directory and the client using generated Bloom filters in order to authenticate themselves. The participating entities must agree beforehand on specific hash functions in order to use these Bloom filters, yet this issue is not resolved but through a static agreement. The scope of the restrictions is very poor compared to our policy solution that provides an efficient semantic expressiveness used to define the security preferences of each entity.

6. Conclusion

In this paper we propose a context aware policy based solution to secure service discovery in Service Oriented Architecture. First we established a threat analysis related to discovery mechanisms by providing a non exhaustive list of possible attacks that can be built against the data and resources of service discovery actors. We tend to cope with these identified threats by defining a secure registry solution relying on context-aware policy. We motivate the use of secure and trusted context-information in order to adapt the security policy enforcement with dynamic environment.

Our approach solves user's privacy and service access control by introducing context-aware access control for discovery service and efficiently supports trust establishment between different actors of the system. We are currently investigating about trust and security management for context acquisition and reasoning.

7. References

[1] OASIS, "UDDI", http://www.uddi.org

[2] WS-Discovery Specifications http://msdn.microsoft.com/ws/2005/04/ws-discovery/

[3] S. Trabelsi, J.C. Pazzaglia and Y. Roudier "Enabling Secure Discovery in a Pervasive Environment" 3rd International Conference on Security in Pervasive Computing (SPC 2006) – York – UK – April 2006

[4] A. K. Dey, "Understanding and using context," Personal and Ubiquitous Computing Journal, vol. 5(1), pp. 4–7, 2001.

[5] R. Liscano and A. Ghavam, "Context Awareness and Service Discovery for Spontaneous Networking", School of Information and Technology and Engineering (SITE), University of Ottawa, 2003

[6] S.E. Czerwinski et al, "An Architecture for a Secure Service Discovery Service" , In Proceedings of MobiCom '99, Seattle, WA, August 1999

[7] Kay Römer, Oliver Kasten, Friedemann Mattern, "Middleware Challenges for Wireless Sensor Networks", ACM Mobile Computing and Communication Review, Vol. 6, No. 4, pp. 59-61, October 2002

[8] J. I. Hong and J. A. Landay, "An architecture for privacy-sensitive ubiquitous computing," in MobiSYS '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services. ACM Press, 2004, pp. 177–189.

[9] N. Shankar and D. Bafanz, "Enabling secure ad-hoc communication using context-aware security services," in UBICOMP 02: Workshop on Security in Ubiquitous Computing, 2002.

[10] L. Bussard L., Roudier Y., "Untraceable secret credentials: Trust establishment with privacy," in PERCOMMW'04. Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004.

[11] J. Van den Bergh and K. Coninx. "Towards integrated design of context-sensitive interactive systems", Mar. 2005.

[12] W3C OWL , "Web Ontology Language". http://www.w3.org/2004/OWL/.

[13] Jess, the Rule Engine for the Java^{TM} Platform, http://herzberg.ca.sandia.gov/jess/.

[14] S.E. Czerwinski et al, "An Architecture for a Secure Service Discovery Service" , In Proceedings of MobiCom '99, Seattle, WA, August 1999

[15] S. Trabelsi, J.C Pazzaglia, Y. Roudier "Secure Web service discovery: overcoming challenges of ubiquitous computing" ECOWS 2006, 4th IEEE European Conference on Web Services, Zurich - Switzerland, December, 2006

[16] F. Zhu, M. Mutka, L. Ni "Prudent exposure: A private and user centric service discovery protocol" Proceedings of the 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom'04) Orlando, USA, 2004

[17] L. Gomez, L. Moraru**,** D. Simplot-Ryl and K. Wrona**,** and**.** Using Sensor and Location Information for Context-Aware Access Control. In *Proc. International Conference on "Computer as a tool" (EUROCON 2005).*