Enabling Secure Discovery in a Pervasive Environment

Slim Trabelsi[*], Jean-Christophe Pazzaglia[+1], Yves Roudier[*]

[*]Institut Eurécom
2229 Route des Crêtes - BP 193
06904 Sophia Antipolis Cedex - France
{Slim.Trabelsi, Yves.Roudier}@eurecom.fr

[+]SAP Labs France
805 Avenue du Dr Donat - BP 1216
06254 Mougins Cedex - France
Jean-Christophe.Pazzaglia@sap.com

Abstract. The pervasive computing paradigm assumes an essentially dynamic model of interaction between devices that also motivates the need to discover the services offered by previously unknown parties at an early phase of these interactions. Whereas this assumption is at the heart of many pervasive computing protocols and systems, the necessity of securing service discovery and the complexity of this task have been largely underestimated, if considered at all. This paper discusses the implications of insecure service discovery in available systems and which security objectives should be pursued. The design space for introducing security features into a specific architecture, namely registry-based discovery systems, is then explored and assessed.

1. Introduction

The Service Oriented Architecture (SOA) paradigm, currently boasted by Web Services, was initially developed in the Jini [13] framework to address the specific requirements of pervasive computing software. In particular, SOA was originally intended to enable access to applications running on nearby devices in a dynamic fashion. This programming style promotes the use of loosely coupled and highly interoperable applications to overstep the limitations of traditional distributed component solutions (CORBA, DCOM). A *Service*, the building block of SOA solutions, is intended for encapsulating a set of related business functions within a container and for enabling access to these functions through standardized interfaces. In pervasive computing, context-awareness would typically be enabled through the access to a set of such services.

Orchestration techniques were developed in order to deploy a set of basic services as a more complex service. Even though static orchestration is frequently used, for example in Web Services based architectures, we argue that the quintessence of the SOA style, especially when used for pervasive computing software, lies in the

[1] New affiliation since submission. Previous affiliation: Institut Eurécom.

[3] To be more precise, this solution uses a middle-agent mechanism often called registry, . depending on its capabilities, this middle-agent is also referred in the literature under the names of directory, discovery service, broker, facilitator,…

dynamic composition of services. Such a dynamic composition obviously comes at a cost: being able to locate previously unknown services becomes mandatory.

Discovery therefore becomes of strategic importance in the SOA stack and this importance is growing proportionally with the dynamic aspect of the environment: while a typical intranet implementation may rely on a basic discovery strategy (e.g. naming service in CORBA) or may even not strictly require it (e.g. predefined set of known services), Internet-wide and, above all, pervasive applications face a set of challenges with respect to discovery. In such applications, the discovery strategy should cope with the heterogeneity of services and platforms from a technical perspective (e.g. take into account bandwidth, energy savings …), with the complex semantics of service descriptions (e.g. resorting to terminology- or ontology-based descriptions), with the scalability of the solution, and with the requirements for security and trust regarding the services discovered. Since the emergence of the SOA paradigm, different works (e.g. UDDI [14], WS-Discovery [3], OWL-S [6]) have aimed at solving many of these issues, yet only very few of them address security and trust, which are however essential to the successful deployment of pervasive computing solutions. For instance, services should protect sensitive information as well as their availability from rogue users; and private information of a user should not be revealed to a service without assessing that service's potential maliciousness.

This paper is organized as follows. Section 2 addresses the relationships between security and service discovery mechanisms. Section 3 discusses different designs that can be incorporated into existent protocols in order to secure them. Section 4 details the security features that should be introduced into a registry based discovery protocol and their handling. Section 5 compares the service discovery model explored in this paper with related work. Open issues not addressed in this paper are finally discussed in conclusion.

2. Service Discovery and Security

A common misconception is to limit service discovery to the registry[3] supported architecture that many standard SOA based services (like for instance Jini's reggie, UDDI, etc.) have adopted de facto in their implementations. In such solutions, as illustrated by Figure 1, the registry is responsible for putting in touch services and their clients. 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. This now rather conventional approach to SOA introduces a third party, which clearly must be trusted.

Service discovery may alternatively rely on a peer-to-peer architecture. Every client then can perform a service discovery by broadcasting its request to its neighborhood, and if one of the neighbors is able to fulfill the request, it will send a response to the requester. The neighbor may otherwise forward this request to its own neighborhood. This mechanism is used for instance by the P2P-based Web Service Discovery system (PWSD) [15], which relies on the Chord P2P protocol to perform

the service discovery. The rest of this paper essentially discusses the registry based model, also called service discovery service.

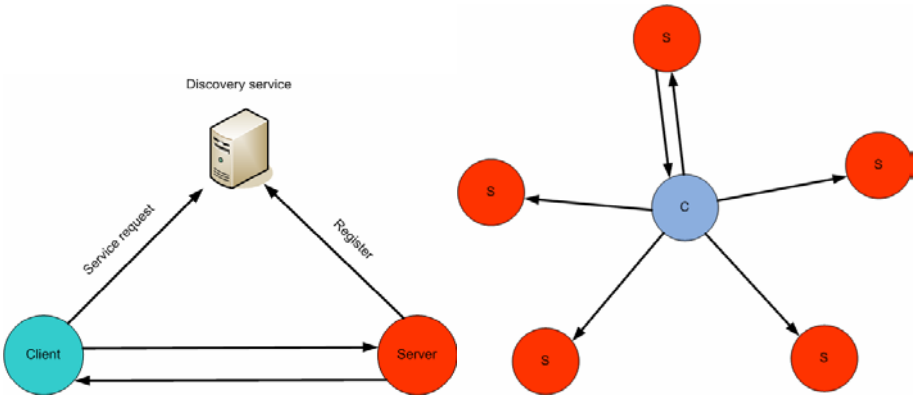

Fig. 1. Middle Agent Service vs. P2P Discovery Architecture

2.1. Revisiting Security Threats for Service Discovery

It should be underlined that the principal actors of the discovery phase are indeed the service requester and the service provider, even in the case of a registry based service discovery. The main specificity of discovery is that, by definition, these actors are initially unaware of their respective existence and of their security policies. In addition, they are likely members of different administrative domains. Discovery is very often at the initiative of the service requester (e.g. *lookup model*) but can also be initiated by the service provider (e.g. *advert model)*. The initiator takes a more important risk than the other party since it does not *control which entities will receive* the discovery message, nor the potential usage of the information embedded in this message. This peculiar situation raises two threats:

- *Client and Service Authentication*: the very objective of service discovery is to communicate with previously unknown entities that provide specific functionalities. Open discovery services therefore require that the first message sent (*lookup* or *advert*) is in clear, also meaning that the content of the message can be accessed. Without the means to authenticate clients and servers, service discovery makes the implementation of a man-in-the-middle attack possible [11], a malicious entity being able to wrongly answer a discovery message. Registry based discovery schemes make it much simpler than infrastructure-less ones to perform secure discoveries, since the registry is the only element which the client needs to identify and which it should be identified from, thereafter enabling message protection through their encryption or integrity check.
- *Privacy*: in the *lookup* model, the information disclosed in the request is likely to reveal a subset of the intentions of the service requester. Similarly, service providers may want to avoid potential commercial competitor or malware to gather

information about their offers too easily. In both cases, attaching an identity certificate to enable the proper identification of parties in presence and the protection of subsequent messages will only expose more information (name, address …). The correlation of such information with discovery related information is particularly worrying from a privacy protection perspective.

In addition to the previous two threats that are relatively specific, the overall purpose of service discovery is to access some resources. This yields more classical threats:

- *Access control*: Since client/service authentication is problematic in the initial discovery phase, traditional service oriented architectures do not support access control during the discovery phase. Service providers would ideally advertise their services exclusively to potential users, even though this objective is in practice difficult to achieve in a pervasive environment. Disclosing the description of a service to any requester potentially increases the risk that a malicious user take advantage of this knowledge to access restricted services.
- *Availability*: Denial of Service (DoS) is an attack to the availability of resources preventing the authorized access to a system resource or delaying system operations and functions. Openly exposing service descriptions during discovery enables attackers to exploit vulnerabilities by creating specially crafted messages for the server or by the registry. Notably, registries clearly constitute a single point of failure and therefore are particularly sensitive to brute force DoS attacks.

2.2. Objectives

With the success of Web Services, the momentum behind Service Oriented Architecture is enormous and thousands of services are already available on the Internet. Similarly, the deployment of wireless networks and small devices (sensors, mobile phones, PDAs, RFIDs, etc.) is likely to boost the startup of local services, thereby paving the way for the actual deployment of pervasive computing systems. In such landscape, where mobility is the usage, clients and servers should protect themselves from malicious software, including at discovery, the very first step of any interaction.

Clients should be able to find a service matching their preferences, these preferences encompassing different characteristics of the service: the service functional definition *per se*, its cost, its quality, as well as the security and privacy requirements imposed by the service (e.g. encryption strength). On the client side, the user should be certain that only services matching his preferences would be chosen and later contacted: from his point of view, trusting a service should therefore go beyond the simple authentication of the service provider and also establish the veracity of the features exposed by 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. Services should therefore be accessible only if they trust the client to access them according to a precise policy that protects them. Matching this policy description is therefore an important part of the discovery process.

Several important concerns should therefore be taken into account in order to enable a secure service discovery service:

- *Security policies for service discovery*
  The different entities participating to service discovery each specify their own discovery objectives as policies. Security policies should be a part of these objectives and should be described using a language common to all entities. Ideally, this language should be expressive and flexible enough to describe standard operations intended to ensure the security of discovery (authentication means, access control, encryption level, etc.) as well as more advanced mechanisms such as auditing or privacy related issues. The matching of client and server discovery policies in general, and security policies in particular, is likely to use reasoning techniques from the Semantic Web area.

- *Responsibility for enforcing the discovery policy*
  Discovery policies can be enforced by the client and by the server. However, discovery policies may also be sent remotely and their enforcement can be delegated to a third party trusted by the client and the service, like the registry. This notably implies that the semantics attached to the policy is non ambiguous.

- *Limiting data exposure during the service discovery process*
  Discovering a service may imply that the user disclose his identity, the kind of service he is looking for, and other personal information like certificates or his location. Even if such information were mainly public, it might contain some data that should not be disclosed to everyone. In order to protect the privacy of the user, a mechanism should guarantee the diffusion and retention of these data. If a service is publicly available and publicized, it will also be more exposed to attacks. A possible way to prevent these attacks is to hide the service, or more precisely to delay its discovery. Comparing services to houses and service methods to doors, it will be far more difficult for a thief to illegally access the house if the doors are hidden and if he does not have any idea about the mechanisms used to open these doors (code, locks …). Servers should similarly have the possibility to choose who can discover their services

- *Establishing trustworthy relationships*
  In pervasive environments, devices and users are likely belonging to different administrative domains. Consequently, an entity cannot take for granted that a discovery related information is true without appropriate proofs. Building secure federations of devices may be required to establish trust on a longer term basis, using extended attribute certificates or even incentive systems (reputation, monetary incentives …).

Next section focuses on describing the adaptation of a registry based protocol enabling it to take into account these concerns and objectives.

3. Secure Service Discovery Model

Assigning to a trusted entity of the system the responsibility to enforce the discovery policies and in particular their security part as defined by users is critical to service discovery. To avoid raising the complexity of service discovery, we do not propose to add a new entity to SOA architectures together with a dedicated protocol, but rather to assign this task to the registry. The reason for choosing the registry as a policy enforcement point is that the users of the system typically already need to trust this entity. This solution thereby implements a distributed security policy and assumes that servers and clients can communicate with the registry through the use of a PKI. (e.g. knowledge of the registry's public certificate) or through some more advanced trust establishment mechanism (which may be more adapted to pervasive computing, see for instance [2]).

3.1. Secure Discovery Service Use Cases

Before detailing the protocol, the policies expressiveness, and the registry capabilities, we will explain in the following use cases how the registry behavior and the messages are modified to fulfill the security requirements imposed by the entities involved in discovery activity. To illustrate these different needs, we take the example of different services offered within a research laboratory. Let us first assume the availability of printing services, whose access should be controlled or monitored due to funding constraints.

The first printing service is a standard black and white printer that is accessible to everybody. This service is advertised to the registry by sending the service description without imposing any particular security constraint. The client will simply send a message containing a service template in order to acquire the exact service description from the registry. This example, shown in Figure 2, introduces the basic discovery protocol, largely inspired by Jini, that we assume to exist and reason about.

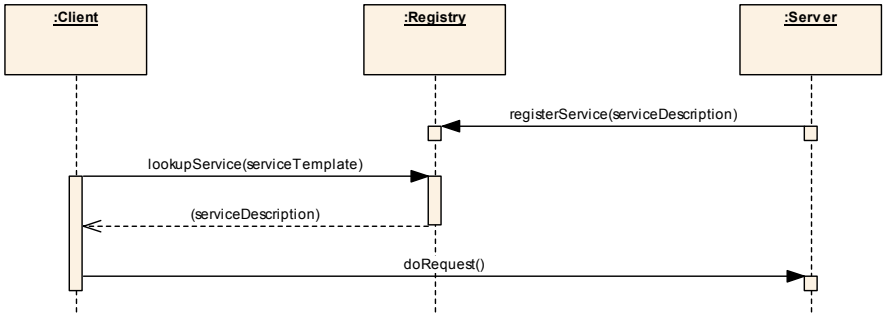

Fig. 2. Basic Service Discovery: the Black and White Printing Service

The second printing service involves a color printer only accessible by the laboratory staff and their guests. The registration message sent by the server should

include a security policy that specifies that the client must be authenticated by the registry and that he must provide a discovery proof signed by the registry to access the service. The client sends his lookup service request by specifying a service template and, in order to authenticate itself, an identity certificate[4]. The registry verifies if the user pertains to the staff or if he is a trusted guest, and then generates a discovery proof (for example a signed SAML assertion). In order to print, the client will have to exhibit this discovery proof within the request as shown in Figure 3.

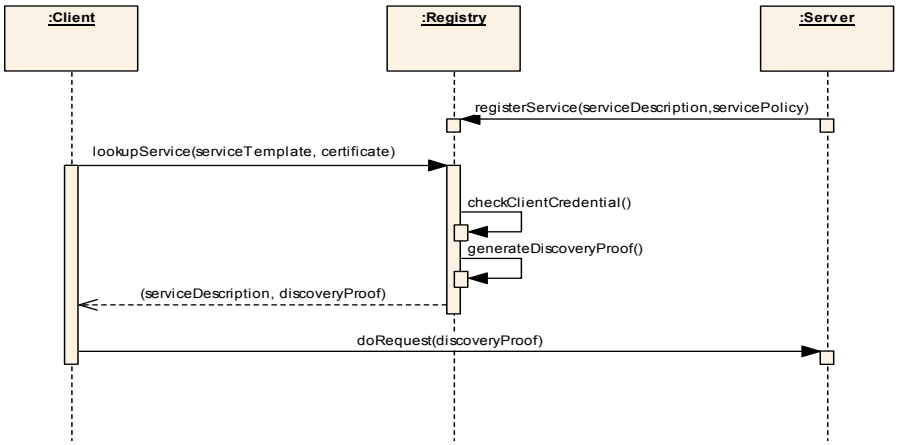

Fig. 3. Secure Service Discovery: the Color Printing Service

The third printing service, a photo printer, is restricted to the Multimedia research team staff. The registration phase is similar to the color printer example: the server advertises itself to the registry by sending a message containing the service description, its security policy requesting the registry to authenticate clients, and also its certificate in order to be authenticated by the registry. The client lookup request contains a service template and a security policy. This policy specifies that the registry has to authenticate the server (to protect the client), while the server policy requires generating a traceable token as shown in Figure 4. This token (for example a SPKI certificate) will be used by the server to authenticate the client while preserving his privacy, yet under certain circumstances, an authorized third party would be able to find the client's identity by contacting the registry.

[4] **NB:** Certificates and proofs are shown as constraints in the UML diagram to provide a condensed representation. These constraints should be enforced through network layer (message over SSL) or using encryption, signature, or metadata embedded in the message (for example SOAP using WSS, XML-DSig, SAML assertions …).

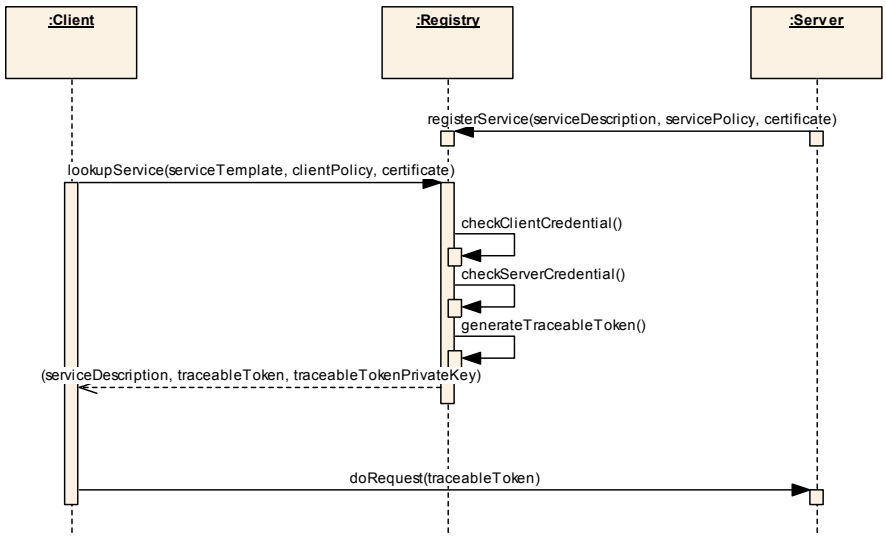

Fig. 4. Secure Service Discovery: the Photo Printing Service

In addition to the printing services, the laboratory also gives users access to two other services: a file sharing service and a backup service. In order to manage file access rights, both services require authenticating the users. In order to enable access to these services, the registry will check that the client belongs to the right user group and also that he is willing to disclose his identity by means of its digital certificate.

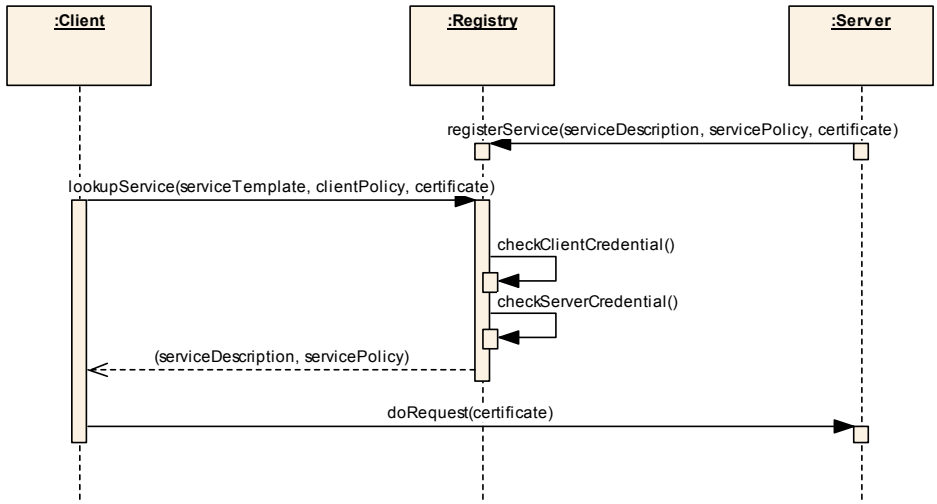

Fig. 5. Secure Service Discovery: the File Sharing Service

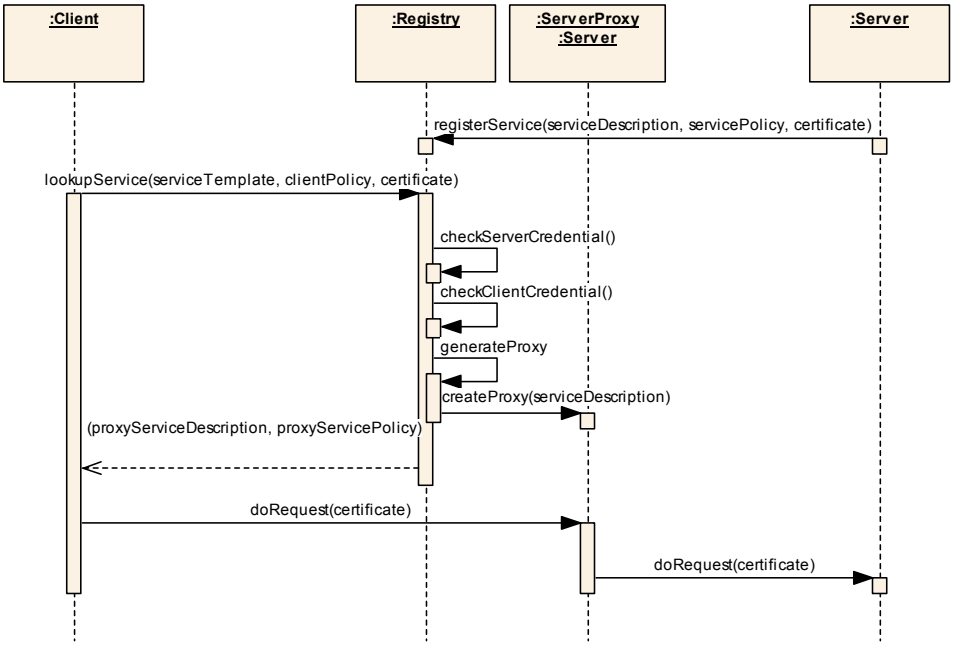

Fig. 6. Secure Service Discovery: the Backup Service

The backup service is working quite similarly to the file sharing service. For security reasons however, the server will not grant access to a client belonging to guests and managed by an external administrator. The access still can be granted provided that the registry generates a temporary proxy within the laboratory's administrative domain. Therefore, instead of sending the service description to the client, the registry will generate a proxy and send the proxy service description. The client will send his request to the proxy which will forward it to the server. In other circumstances, the proxy generation may also be triggered by the client in order to better protect its privacy. In this case, the proxy could authenticate the clients but only exhibit traceable credentials to the server like in the photo printing services described before.

4. Detailing the Protocol and Registry

Standard service discovery schemes do not enable the clients and the servers to impose their own security policy during the *lookup* phase. The originality of our solution is that we consider that service discovery transactions have to satisfy the security preferences of both entities by design. To fulfill this requirement, each client and server should export its security policy using a common language with well-defined and non ambiguous semantics. As mentioned earlier, the registry must be trusted by all the entities of the system since the security policy enforcement during

discovery will be delegated to it. As we rely on a PKI infrastructure to authenticate this registry in our examples, the different clients and services must know the registry's public key to protect discovery messages.

4.1. Security Levels in the Discovery Policy

Broadly speaking, security policies are meant to define the reciprocal commitments of the partners involved in a specific process. Their coverage usually depends on the system security objectives (confidentiality, privacy, access control, availability…) and on the degree of expressiveness of the security policy language (e.g. from a low level of expressiveness like with WS-Policy [5] to a high level like with Rei [4]). In the case of service discovery, the scope of the policies may vary: they can be restricted to the service discovery process or they can impose constraints to the service execution or access. Three levels of policy scopes should be distinguished:

- Level 0: No security policy definition; the user must have the possibility to perform a simple and non secure service discovery. In that case, the different entities do not need to specify a particular security policy.
- Level 1: A security policy restricted to the service discovery aspects. The servers set up a security policy to limit their visibility to some restricted clients. Symmetrically, the clients may limit the scope of diffusion of their request to some particular servers. These policies will be enforced by the registry and they do not apply to the future transactions between the client and the service.
- Level 2: A security policy that specifies requirements of the service discovery step and also specifies how to configure access control to the service. Indeed, being able to discover a service does not necessary imply being able to effectively, even less securely access this service if one does not know the proper requirements. The policy rules can also force entities to set security associations (like the choice of the encryption key size or the encryption algorithm) or may imply that the client exhibit generic or specific credentials generated during the service discovery process. A level 2 policy may specify that the registry should take care of forwarding service requests after service discovery because of privacy requirements in the policy.

To our knowledge, no security policy language specifically is adapted to service discovery or enables expressing foreseen privacy or traceability requirements, even though inspiration might be taken out of e.g. EPAL or P3P. In our model, the registry primarily acts as a remote policy decision point (ensuring that client and server policies are compatible) but will also ensure the enforcement of privacy or traceability requirements. Still, the client or server may alternately implement and commit to the enforcement of other counterpart requirements or leave it to the registry. Level 2 policies also imply a tight integration with service access security mechanisms.

4.2. Service Registration: registerService

In order to register its service, the service provider advertises its capabilities to the registry. A `registerService` message should embed the following information:

- Service properties: this field contains the description of the service capabilities. This description will be used during the matchmaking process while looking for a suitable service provider for a specific client.
- A security policy: this field contains all the security preferences of the server. It can be considered as a set of access control rules that must be satisfied by the client to discover and access its proposed services. Such policies may be exchanged by using a standard protocol for metadata exchange (e.g. WS-MetadataExchange).
- A certificate: this certificate is signed by a trusted certification authority and contains certified information about the server (identity, public key, the lifetime of the service, recommendations …). The information embedded in this certificate can be used by the registry to verify that the server complies with the security constraints imposed by the service requestor.

The second and third components of the message are optional and vary with the expected security level. A service is public by default: if the server does not have the need nor the capabilities to protect its services, it does not have the obligation to specify a security policy. The server's certificate presence is optional for service discovery, but mandatory to enable its secure discovery. Moreover, its absence may restrict the potential constraints that clients may impose on the server's credentials or attributes described in the certificate.

4.3. Service Discovery: lookupService

In order to obtain a list of suitable services, the client must describe its service preferences and provide its credentials. Preferences will be sent to the registry, which will search for the services fitting with these requirements. The `lookupService` message should thus contain:

- Service template: this template is based on a partial instantiation of a service description constraining certain aspects of the service. This template may correspond with some existing services assuming that all the entities share a common taxonomy for instance. The registry would then match this template with the stored description of available services.
- A security policy: this policy contains the security constraints required by the client. These constraints describe the properties that must be satisfied by the service provider. These constraints may impose to enforce server authentication, to choose servers from a restricted set of administrative domains, or to check that the server respect precise rules concerning data retention, for example.
- A certificate: this certificate should be signed by a trusted certification authority and should contain certified information about the client (identity, public key,

role…). This certificate can be used by the registry to verify that the client corresponds to the security criteria of a server that seems to match a lookup query.

Similarly to service registration, the policy and certificate are optional. This will obviously impose symmetrical limitations on the client's ability to locate services.

4.4. The Registry Matching and Policy Enforcement

The discovery process is initiated by the client sending a `lookupService` message to the registry. The registry will perform the following actions:

- Trust establishment and public key distribution: This bootstraps secure service discovery. The registry is considered as a trusted third party, and in order to establish this trust every entity that joins the system must acquire the public key certificate of the registry to identify it and to secure their communication.
- Match the service template contained in the client's message with the description of the registered services.
- Filter out the services that advertise a security policy incompatible with the client credentials or incompatible with the client's security policy.

In order to conform to the server and client policies, the registry may generate a suitable token such as a proof of discovery, an authentication receipt, or a temporary certificate, and associated keys to access the service. Finally, the registry returns the client a list of service descriptions answering his request, possibly with the token and server security policy that must be complied with to access to the proposed services.

4.5. Model discussion

As explained in Section 2, challenges in securing service discovery cover the broad spectrum of security issues. Integrating security policies on top of a registry-based architecture solves these following concerns in this way:

- Authentication: each entity may require the other entities to authenticate themselves to the registry during the registry/discovery phase. The description of required authentication means during service delivery can also be specified using a level 2 policy (used during service matching and enforced later).
- Confidentiality: by using a PKI infrastructure to secure the transactions between the registries and other entities and by also using the security tokens to secure the communications between entities, only authorized entities can access messages.
- Access control: Server resources are protected against an unauthorized discovery thanks to security policies; furthermore the registry is able to enforce clients' requirements during service discovery by selecting only services providing valid credentials or secure enough access mechanisms.

- Trust: a PKI based solution establishes a trustworthy relationship with the registry. This trust relies on standard certification and trust chain. Further trust assessment will be performed by the registry itself. Although this can be seen as a limitation in fully ubiquitous environment, this solution is sufficiently flexible to enable the mobility of users across different administrative domains.
- Privacy: each entity can selectively expose its services to a restricted set of entities while remaining hidden for the others. Contrarily to existing solutions, our model does not focus only on servers, but also on clients and their requirements.
- Non repudiation: thanks to security tokens generated by the registry, an entity has the possibility to prove that it discovered the system legitimately. Access to this proof may also be controlled by policies to comply with a legislative framework.

This solution is strongly dependent on the existence of a trusted authority that is in charge of enforcing the security policies of the different entities in the system. To establish this trusted relationship, we also make an assumption concerning the *a priori* knowledge of the registry's identity or public key (or any kind of certificate) by services requestors and providers. These two constraints make the solution difficult to deploy in some conditions (the registry is missing or not reachable, or the users cannot securely acquire the registry certificate).

5. Related Work

Discovery-enabled pervasive computing systems generally address access control when a service is accessed, and make no restriction whatsoever regarding service discovery itself. In addition to privacy issues, this approach requires the application programmer to understand and address security issues at the implementation level instead of specifying an abstract and probably less error-prone security policy.

In contrast, [7] details the architecture of a service discovery service that addresses security at the discovery stage, as discussed in this paper. This work focuses on protecting the service side against malicious parties through strong authentication of end-points and the client side against private information disclosure and man-in-the-middle attacks through authentication and encryption. The system also supports the specification of a rudimentary discovery policy making it possible for a client to restrict the discovery to the set of services run by a trusted entity. However, availability and entity privacy are not addressed in this work.

The authors of [9] propose an architecture for secure service discovery. Components share a multicast address that will be used to bootstrap the communication. Directories (i.e. registries) use this multicast address to periodically announce their unicast address and certificate. Proxies are used to protect the servers by handling the registration, authentication, authorization, and key management for them, entities in the system setting up a session using hybrid encryption. The number of proxies (one by service) and heavy PKI infrastructure for securing the communication are likely to generate an important message overhead. Contrary to the claimed objective of this work to address pervasive systems, services are likely to be

static, whereas the approach we advocate only requires the availability of a local registry, each client potentially being a server for other clients.

[12] considers privacy issues during service discovery. The proposal is based on the use of bloom filters to protect the client and server personal information (identity, certificates, attributes…) during directory and client authentication. However, the participants must agree in advance on the hash functions used. In comparison, the present paper's approach to privacy is quite different since the scenario it addresses relies on the existence of a trusted party for service discovery, whose role is only slightly extended to the non-disclosure of private information.

SPDP [8] is a discovery protocol for infrastructure-less and in particular registry-less systems, a much more complex situation than the one addressed in this paper. The design of this protocol is also strongly influenced by practical considerations regarding the electric consumption of mobile devices. SPDP however relies on a web-of-trust model to decide which devices may participate to the network and consequently to the discovery. This model however cannot delegate the enforcement of a discovery policy. In comparison, the approach presented in this paper relies on a TTP and a PKI, yet it still preserves a reasonable openness, if one considers that the scenarios depicted generally assume the availability of a trusted and known entity.

6. Conclusion

Service oriented architectures were originally introduced to address the deployment of software in a pervasive computing context. Even though they have been studied for some time now, the security of service discovery, one of their essential features, is generally weak. This paper analyzed such threats in SOA stacks that use a registry supported discovery and showed that discovery exhibits specific security needs, notably due to its early occurrence in the interaction of two devices. We proposed to reuse the messages of a standard discovery protocol by only extending their contents with the security information required to secure discovery and by delegating the handling of such information to the registry.

The solution proposed aims at dynamically configuring the mechanisms for protecting the security and privacy of the server and of the service requestor. It requires a trusted infrastructure for discovery, namely the registry, which plays a key role by ensuring that the discovery policies specified by the service and by the requestor are matched in a trustworthy manner. This prerequisite may limit the usage of the discovery mechanism in completely ad-hoc situations yet it is well adapted to the deployment of pervasive applications at airports, public agencies' or private companies' premises for instance. The discovery policies mentioned above raise some open issues regarding the fusion between client and service requirements: Semantic Web mechanisms for service description and matchmaking [6] will probably need to be adapted in order to be able to describe and negotiate security aspects of policies.

We are currently implementing the solution described above as part of the middleware developed within the European project MOSQUITO[5] [10]. In addition to that solution, we are currently investigating an alternative infrastructure-less solution for service discovery which should still provide a significant subset of the security properties enforced by registry based service discovery.

Bibliography

1. F. Zhu, M. Mutka, and L. Ni "Classification of Service Discovery in Pervasive Computing Environments," MSU-CSE-02-24, Michigan State university, East Lansing, 2002.
2. L. Bussard, Y. Roudier "Embedding Distance Bounding Protocols within Intuitive Interactions" 1st International Conference on Security in Pervasive Computing, SPC'2003, Boppard, Germany, March 12-13, 2003.
3. OASIS, "WS-Discovery" http://msdn.microsoft.com/ws/2004/10/ws-discovery/
4. L. Kagal "Rei: A Policy Language for the Me-Centric Project" HP Technical Report, 2002
5. OASIS, "WS-Policy Specifications" http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnglobspec/html/ws-policy1202.asp
6. D. Martin et al, "Bringing Semantics to Web Services: The OWL-S Approach", Proceedings of the 1st SWSWPC, USA 2004.
7. S. Czerwinski, et al "An architecture for a Secure Service Discovery Service" Proceedings of Mobicom'99, Seattle, USA, 1999
8. F. Almenarez, C. Campo: "SPDP: A Secure Service Discovery Protocol for Ad-hoc Networks", In 9th Open European Summer School and IFIP Workshop on Next Generation Networks, Budapest, 2003.
9. F. Zhu, M. Mutka, and L. Ni, "Splendor: A secure, private, and location-aware service discovery protocol supporting mobile services", in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (Percom'03)*. IEEE Computer Society, Mar. 2003, pp. 235–242.
10. MOSQUITO Project IST 004636 https://www.mosquito-online.org/
11. M. Ghader *et al* "Secure resource and service discovery in personal networks" Wireless World Research Forum Meeting #12, Canada, 4-5 Nov 2004
12. 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
13. SUN Microsystems, Jini Specifications http://java.sun.com/products/jini/
14. OASIS, "UDDI", http://www.uddi.org
15. O. Garofalakis et al "Web Service Discovery Mechanisms: Looking for a Needle in a Haystack?" 15th ACM Conference on Hypertext and Hypermedia (Hypertext 2004), Santa Cruz, USA, 2004

[5] The FP6 IST project MOSQUITO is supported by the European Community. This document does not represent the opinion of the European Community. It is also the sole responsibility of the author and not the responsibility of the European Community using any data that might appear therein.