Trust protocol integrating services' semantics

Cédric Lévy-Bencheton, Frédéric Le Mouël

Trust protocol integrating services' semantics

Cédric Lévy-Bencheton, Frédéric Le Mouël
{cedric.levy-bencheton, frederic.le-mouel}@insa-lyon.fr

INRIA ARES/CITI, INSA-Lyon, F-69621 FRANCE

Abstract. In high dynamic environments such as Internet or pervasive computing, different unknown services are provided. When two services communicate, they share properties to negotiate a provided service. By introducing trust, services rely on the identity of the participants, to evaluate a trust level and adapt this provided service. However, it is not always possible to verify identity. In this paper, we propose a service-to-service trust protocol based on services' description. By exchanging their descriptions, services build trust in each other and adapt the quality of the provided service through policies.[1]

1 Introduction

In high dynamic environments, services communicate and share information to determine a provided service. This process is done with no previous knowledge of each other. By introducing trust, based on the service's identity, a service can adapt the quality of its provided service. Since identity verification is not always available, a new trust model is needed. In this paper we introduce a service-to-service trust protocol based on the services' description. Two services communicate and trust each other on disclosed properties, representing a service's description. The description is dynamic according to the trust level: properties can be disclosed or hidden. Then, the provided service is configured according to the evaluated trust level from one service in another. The trust becomes part of the services' life-cycle.

In Section 2 we expose different problems on current trust and services implementations, in Section 3 we present a service-to-service trust protocol, in Section 4 we present different scenarios implementing our model.

2 Related Works

As we already supposed, an identity-based trust is not always verifiable. Current approaches rely on certificates, reputation, a machine name/address or a chip to verify identity [3].

A trust is a value representing an expected action, influenced by the context, previous exchanges or the relationship between two participants [4]. Combinations are possibles. Trust is able to make decisions or discover new services depending on different rules, described in policies [5]. However, in previous works,

[1] This work is supported in part by the ACI KAA [1] and by the ARC PRIAM [2].

the trust level grants/refuses access to a provided service [6, 7]. We propose to adapt this provided service according to the trust level.

A service can be defined as a behaviour described by its semantic description [8, 9]. The discovery of services allows to find a wanted service using its description [10]. The negotiation leads both sides to exchange properties or sign a contrat in order to configure the provided service [11]. In classical services approaches, trust in the requester (asking a service) or the provider (providing a service) does influence the discovery or negotiation. In [12], an identity-based trust, using services' semantics, influences those process, but relies on a trusted third-party to ensure identity.

In a Service-to-Service environment, the trust evaluation occurs directly between two services. Thus, identity-based trust is not always possible. As previously said, the service's description represents the service's behaviour defined by its properties. By introducing a description-based trust, it becomes possible to configure a provided service and gain trust by relying on this description. We propose to build trust through the service's description. We enhance the services discovery with trust propagation to find trusted services. Then we adapt the provided service during the negotiation, by disclosing or hiding properties of the description, thus raising or lowering trust. In the next Section, we present our service-to-service trust protocol.

3 A Service-to-Service Trust Protocol

3.1 A Description-based Protocol

We present a trust protocol based on the services' description. In our approach, both services trust each other by sending their descriptions. This service-to-service protocol evaluates trust with no need of third-parties or certificates. The service's description defines the service's behaviour. A description contains a list of properties (i.e. the service's "name", "type" or "creator"). A property has a *name*, and a couple (*value*,*trust value*). The trust value is determined by the service, and discloses a property when the service trust in another service is above the associated trust value. Our protocol expects a service to send its description in order to be trusted. Hence, the service's description is filtered: only properties available for an evaluated trust level are sent. The trust evaluation and the description filtering are done by policies defined by the service.

A service uses the *trust evaluation policy* to evaluate trust in a description by looking at the different properties and their values. The trust is evaluated via a matching policy: for every property, a score is given to its semantic value. The resulting trust is the addition of all properties' scores. The matching policy is supplied by the service, who associates a numerical value to semantic values of a property. The service expects others services to have certains properties (such as the same creator, a common type or a defined name), and gives values high enough to the matching for those properties. This way, the value supplied by the trust evaluation policy allows the service to disclose new properties. In order to

simplify this process, services creators should use a standard ontology. The choice of such an ontology is out of the scope of this paper. Then, the service filters its own description using the *filter policy*. This policy takes a service's description with trust values and a given trust level to return a *filtered description*. A filtered description is a description with no trust values: only properties and values are disclosed.

Consequently, a service should defined its own properties trust values in parallel to its policies. A trust value for a property must take into account the different way to gain trust in the trust evaluation policy. This way, a service only discloses the property to trusted services with expected properties in their description, while keeping it secret for untrusted services.

In this protocol, two service roles exist: the *requester* and the *provider*. A requester starts the protocol by looking for a wanted-service, and contacts a provider. A provider is a service answering to a requester, and can become a requester in composed services. The requester and the provider represent the two sides of the communication. The description based approach allows us to adapt the behaviour of the provided service according to the trust level.

3.2 Protocol Steps

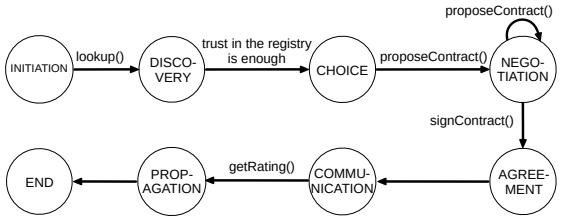

Fig. 1. The Protocol Steps

The different steps of the protocol are represented on Fig. 1. The transitions between the different states are decided by policies. The different messages exchanges are represented on Fig. 2.

The **Initiation** step is the first step. At the initiation, the *requester* filters its own description on a *null* description, so that a description for an unknown service is obtained. The requester also creates a description of a *wanted-service*, with no trust values.

At the **Discovery**, the requester sends a message containing those two descriptions to a registry in order to find a provider for the wanted-service. The registry is a service allowing providers to publish their descriptions. It runs either on a machine, listing all its available services, or on a dedicated server. The registry stores providers' descriptions, with the trust values, and evaluates trust in the requester's description. If the trust is enough, the registry filters its own

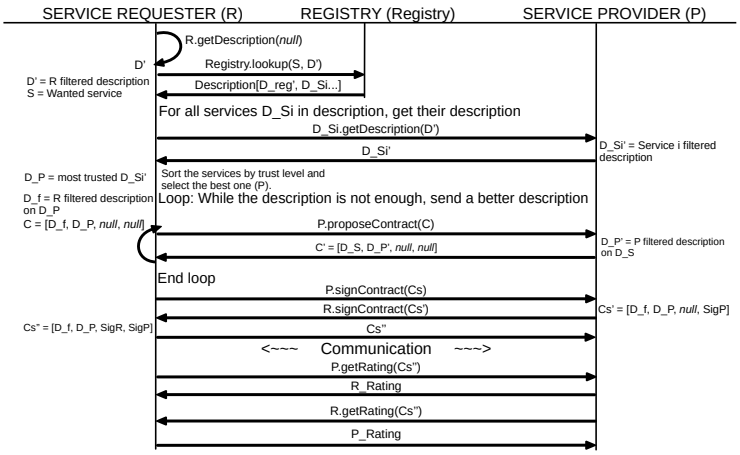

Fig. 2. Protocol

description, and looks up for matching providers. Those descriptions are filtered by the registry using its trust level in the requester. The requester receives the registry's and provider's filtered descriptions and evaluates the trust in the registry. If the trust level is enough, it moves to the next step. On the contrary, the requester tries to find another registry.

At the **Choice**, the requester must select a provider from the received list. The requester trusts the registry on its description, but might not trust the received providers descriptions. To remove this doubt, the requester sends its filtered description for an unknown service to every provider. The providers evaluate trust in the requester and send their filtered description. When the requester has received all descriptions, the most trusted one becomes the provider, in case this description is not below the *minimum trust level* set by the requester. The requester filters a new description based on the provider's description. If no provider is chosen, the requester must change the wanted-service or find a new registry.

At the **Negotiation**, the requester and the provider exchange a contract containing the requester's filtered description, the provider's filtered description, the requester's signature and the provider's signature, in that order. The *trust in a contract* is the trust a service has in the description of the other side of the contract. The requester proposes a contract to the provider containing the requester's new filtered description, the selected provider's description, and *null* signatures. The provider evaluates trust in the requester's description, and sends back a contract. During this process, different policies are called. The *verify contract* policy verifies that the service's description in a contract is the same as the latest description sent, and that the other side's description contains new properties. To this mean, both sides must keep the latest sent and received description. However, a service can decide to not disclose anything new, because the trust on the other side is not enough. In this case, the other side must take

risks and disclose new properties. The *temporary trust increase* policy temporary raises the trust level in a service by a certain value. This policy aims at providing a better description and gain trust. But the other side must not gain too much information without providing anything. The *maximum number of trust increase* policy specifies the number of times a service is allowed to increase its description while the other side does not disclose anything new. When this number is reached, the communication is stopped and the requester must find another provider. The two services keep exchanging contracts until the trust in the contract is above the *trust acceptation level* and move to the agreement.

At the **Agreement**, the requester sends the contract to the provider. The provider checks if the contract is the same: the descriptions are the same as the last sent and received. If it is not the case, the contract is rejected. Otherwise, the provider signs the contract and sends it back to the requester, which does the same and sends a double-signed contract to the provider. Both sides store the contract for a later use.

At the **Communication** step, the requester uses the provided service.

The **Propagation** introduces a *Rating*, a contract with an associated trust value. A service rates a contract by adding its trust in the other side. The trust depends on respect of the contract during the communication. If one service respects its part of the contract, the contract is trusted by the other service. The requester asks the provider to rate the contract. The provider checks that the contract is the same, evaluates trust in the requester and adds this trust value to the contract to create a rating. The rating is sent back to the requester which can store it, and vice-versa. A service stores a rating in its description, under the property "history", with an associated trust value representing its trust in the other side of the contract. A requester stores the provider's rating in its "history" with an associated trust value representing its trust in the provider. Ratings can later be propagated to other services with the same trust level, to show proofs of previous exchanges.

4 Implementation: Scenario Illustration

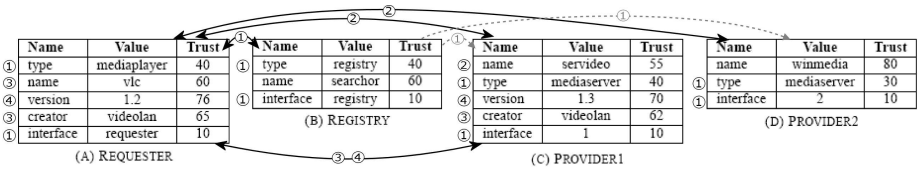

Fig. 3. The Different Services during the Scenarios

We implement four services in JAVA using the OSGi platform: one requester, two providers, and a registry. Their description with the trust values are represented on Fig. 3. The registry stores the providers' description with trust values.

In our implementation, the registry runs on a centralized server, but this does not influence the way the protocol works. We present three different scenarios with different trust acceptation values: *low* with a direct acceptation, *medium* with an accepted negotiation, and *high* with a refused negotiation. The contract acceptation is done by the requester only, to show the importance of the policies in the negotiation phase. The different phases and the disclosed properties are represented on Fig. 3 by circled numbers.

To start the protocol, the requester filters its description for an unknown service. Its policy specified that an unknown service is trusted at 50%. The requester discloses all properties below in its description: "type mediaplayer" and "interface requester" (① Fig. 3(A)). The requester builds a wanted service: "type mediaserver". The filtered description and the wanted service are sent to the registry to find a service provider.

The service registry evaluates trust in the requester to 50%, according to its trust evaluation policy, and filters its description accordingly (① Fig. 3(B)). The providers descriptions are also filtered with that trust level (① Fig. 3(C),(D)). The requester receives the registry's description ("type registry" and "interface registry"): its trust is above the minimum trust level, defined at 45%. The requester moves to the choice. Both providers' descriptions contain their "interface" and "type mediaplayer".

The requester contacts the two providers on their interface with its filtered description on an unknown service (② Fig. 3). They evaluate trust to 60% and return their filtered description: Provider1 discloses "name servideo", Provider2 nothing more. The requester receives the descriptions and evaluates trust in Provider1 to 70%, in Provider2 to 60%. Provider1 is selected to be the provider.

The negotiation starts and the requester creates a new contract, by filtering its description on the provider's received description: "name vlc" and "creator videolan" are disclosed (③ Fig. 3(A)). The contract is sent to the provider, which evaluates trust in the contract, i.e. the trust in the requester's description, to 70%. The provider discloses "creator videolan", which needs 62% to be disclosed, and adds this new description to the contract (③ Fig. 3(C)). The requester receives the contract and evaluates trust to 70%. In scenario 1, the contract is accepted and the exchange moves to the agreement. In scenario 2, the contract is not accepted: the requester builds a new contract, but no new properties are disclosed on either side. The requester temporary raises its trust level by 5%, as defined in its policy, in order to obtain a new description. The same contract is received again: the requester increases its trust by 5% again and keeps in memory it is the second time. The requester has a temporary trust of 80% and discloses "version 1.2" (④ Fig. 3(A)). The provider has now enough trust to disclose "version 1.3" (④ Fig. 3(C)). The requester receives the contract and accepts it in scenario 2. In scenario 3, the trust is not enough: the trust increase process happens again. However the same description is received three times: the requester stops the communication.

At the agreement the requester asks the provider to sign the contract. The provider checks that the contract is the same, signs it and asks the requester to

sign it. The requester does the same and sends the contract to the provider. The services communicate.

At the propagation, the requester asks the provider to rate the contract. The provider verifies that the contract is the same, and rates the contract with its trust in the contract, i.e. its trust in the requester. The rating is sent to the requester which stores the rating in its description under the "history" property, with its trust in the contract as the associated trust value. The same process happens when the requester asks a rating to the provider.

5 Conclusion and Future Work

This paper proposes a service-to-service trust protocol using the services description. This approach does not rely on a third-party service, and does not need to know the device's or user's identity. The services use a trust-enhanced description, filtered according to the trust level in another services' description. Services trust each other on their semantics. Two services need to trust each other in order to access a service. The trust is gained by disclosing more properties of their description. This way, only services trusting each other can communicate.

In future work, the whole protocol should be implemented in the registry, using the negotiation, agreement, and rating. Different tools must be developed to calibrate the trust values to use in the service and in policies. The "history" property should be used to raise trust, using recommendations. Also, the security of this protocol has not been studied, we rely yet on a secured channel, but we should evaluate it before a real use.

References

1. French Ministry ACI Project: Knowledge Authentification Ambient (KAA) (2007) http://kaa.citi.insa-lyon.fr/.
2. INRIA ARC Project: Privacy Issues in Ambient Intelligence (PRIAM) (2007) http://priam.citi.insa-lyon.fr/.
3. Trusted Computing Group: Trusted Platform Module (2005)
4. Artz, D., Gil, Y.: A Survey of Trust in Computer Science and the Semantic Web. Web Semantic **5**(2) (2007) 58–71
5. Sharmin, M., Ahamed, S.I., Ahmed, S., Li, H.: SSRD+: A Privacy-aware Trust and Security Model for Resource Discovery in Pervasive Computing Environment. In: Proc. of the 13th COMPSAC, IEEE Computer Society (2006)
6. Nejdl, W., Olmedilla, D., Winslett, M.: PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web. In: Proc. of the SDM VLDB 2004. Volume 3178 of Lecture Notes in Computer Science., Springer (2004)
7. Ryutov, T., Zhou, L., Neuman, B.C., Leithead, T., Seamons, K.E.: Adaptive Trust Negotiation and Access Control. In: Proc. of the 10th ACM SACMAT. (2005)
8. David Martin et al.: OWL-S: Semantic Markup for Web Services (2004)
9. D. Roman et al.: Web Service Modeling Ontology. Applied Ontology **1**(1) (2005) 77–106

10. Marin-Perianu, R.S., Hartel, P.H., Scholten, J.: A Classification of Service Discovery Protocols. Technical Report TR-CTIT-05-25, University of Twente, Enschede (2005)
11. Lock, R.: Automated Negotiation for Service Contracts. In: Proc. of the 13th COMPSAC. (2006)
12. Galizia, S.: WSTO: A Classification-Based Ontology for Managing Trust in Semantic Web Services. In: Proc. of the 3rd ESWC. Volume 4011 of Lecture Notes in Computer Science., Springer (2006)