Future Generation Computer Systems 55 (2016) 176–185

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Matchmaking semantic security policies in heterogeneous clouds

Giuseppe Di Modica, Orazio Tomarchio *

Department of Electrical, Electronic and Computer Engineering, University of Catania, Viale A. Doria, 6 - 95125 Catania, Italy

HIGHLIGHTS

- Semantic matchmaking of security policies in cloud environments.
- Security ontology for modeling security concepts.
- Automatic semantic annotation of WS-SecurityPolicy policies.

ARTICLE INFO

Article history:
Received 30 June 2014
Received in revised form
11 January 2015
Accepted 9 March 2015
Available online 23 March 2015

Keywords:
Cloud computing
Security policies
Semantic
Ontology

ABSTRACT

The adoption of the cloud paradigm to access IT resources and services has posed many security issues which need to be cared of. Security becomes even a much bigger concern when services built on top of many commercial clouds have to interoperate. Among others, the value of the service delivered to end customers is strongly affected by the security of network which providers are able to build in typical SOA contexts. Currently, every provider advertises its own security strategy by means of proprietary policies, which are sometimes ambiguous and very often address the security problem from a non-uniform perspective. Even policies expressed in standardized languages do not appear to fit a dynamic scenario like the SOA's, where services need to be sought and composed on the fly in a way that is compatible with the end-to-end security requirements. We then propose an approach that leverages on the semantic technology to enrich standardized security policies with an ad-hoc content. The semantic annotation of policies enables machine reasoning which is then used for both the discovery and the composition of security-enabled services. In the presented approach the semantic enrichment of policies is enforced by an automatic procedure. We further developed a semantic framework capable of matchmaking in a smart way security capabilities of providers and security requirements of customers, and tested it on a use case scenario.

1. Introduction

Cloud computing technology has definitely reached the commercialization phase of development. Several commercial cloud providers offer a variety of services, ranging from the IaaS to the SaaS, which users can access to build up cloud services for disparate purposes. However, if on the one hand the provision of computing resources as if they were a utility (such as the electricity) is potentially revolutionary as a computing service, on the other one presents many major problems on information policy, including privacy, security, reliability and access control issues [1].

High-level security constraints about the usage and the access to cloud services and resources should be logically separated from the service itself, and expressed in such a way to be interoperable among the different cloud providers' low level security mechanisms. This would enable secure scenarios of world-wide and cross-domain interoperability among services, in line with the perspective of a services *ecosystem* fostered by the SOA paradigm. One of the approaches adopted to get a secure environment of interoperable cloud services relies on cloud federations. In a cloud federation an individual customer is granted the same level of security, no matter the specific cloud system he decides to access. Though profitable to the customer, this approach requires that the participating cloud systems reach a common mutual trust, therefore the end-user's security will be affected by the trust level eventually established among providers. If cloud federations have reached consensus in academic contexts, there is skepticism that the business model beneath it will appeal commercial players. The approach we support, and that is source of inspiration of this work, is instead to put the end-user in the condition of building up a custom security environment, i.e., an environment which is compatible with their

* Corresponding author.
E-mail addresses: Giuseppe.DiModica@dieei.unict.it (G. Di Modica), Orazio.Tomarchio@dieei.unict.it (O. Tomarchio).

http://dx.doi.org/10.1016/j.future.2015.03.008
0167-739X/

specific security requirements. The basic idea is to put in the end-user's hand a tool to discover which of the available and heterogeneous cloud services match both functional and security criteria; with those services the end-user will be able to build up their secure service context.

Today, the main approach followed to express high-level security constraints is based on the usage of metadata and languages for the specification of security policies [2]. It allows to abstract from the low-level security mechanisms of the specific provider's infrastructure. The benefit of using languages for the specification of security policies is twofold. On the one hand the providers are able to advertise the security capabilities implemented within their administrative domain; on the other one the customers can easily express the security requirements they need for their applications.

But, if with respect to the message security there are several well established techniques and mechanisms, the discovery and compatibility of security requirements among "interoperable services" still lacks of an established methodology. In order to ensure security in all the possible scenarios that may arise, mechanisms must be devised to dynamically verify the compatibility between the consumer's security requirements and the service provider's security capabilities.

The approach we propose calls on well-known policy and security specifications. We define customers' and providers' security requirements/capabilities within policies which are compliant to well-established specifications, such as WS-SecurityPolicy [3]. As it will be clear further in the paper, one drawback of WS-SecurityPolicy is that it only provides for syntactic matching of security policies. In fact, security policy matching depends on the policy intersection mechanism provided by WS-Policy [4]. What characterizes our approach is that security requirements and capabilities are *semantically* enriched, thus enabling the employment of smarter mechanisms to make the match between what is required and offered in terms of security, respectively on the customer and on the provider side. The contribution of this work, which extends that presented in [5], consists in the design of an ontology defining main security concepts, a procedure which automatically maps syntactic security requirements into semantically-enriched concepts, and a matchmaker engine which is capable of reasoning on the customer's and the provider's security requirements/capabilities to derive a match level.

The rest of the paper is structured as follows. Section 2 presents background and related work. Then in Section 3 the overall architecture of our framework is presented. Section 4 presents the semantic model we designed. In particular, we thoroughly discuss the developed security ontology, provide details on the mapping procedure and on the policy matching algorithm. In Section 5 two example case studies are reported to validate our approach. Finally we conclude the work in Section 6.

2. Related work

The adoption of a policy based-approach for managing a system requires an appropriate language for policy representation and modeling and the design and development of a policy management framework for policy matching and enforcement. Several policy languages and framework have been developed following different approaches, and sometimes have been proposed in different application domains [6]. Among the most notable efforts in this domain, worth citing are Ponder [7], a declarative object-oriented language that supports the specification of several types of management policies for distributed object systems, Kaos [8], a policy management framework where concepts and policies themselves are represented using OWL, and Rei [9], a policy framework where OWL is extended with the expression of relations like role-value maps, making the language more expressive than the original OWL. Kaos and Rei are ontology based policy languages that, although not specifically focused on the security domain, are able to express complex security policies: policy matching is also supported by the advanced reasoning capabilities these languages offer. However, when dealing with enterprises and cloud systems that adopt a service oriented paradigm, we believe that an approach compatible with existing standard for policy specification should be followed as far as possible.

WS-Policy [4] is the specification used in service oriented architectures to express policies. A policy is defined as a collection of alternatives and each alternative is a collection of assertions. Assertions specify characteristics that can be used for service selection such as requirements, capabilities or behaviors of a Web service. Policy assertions can represent both requirements on service requesters and capabilities of the service itself. Requirements represent a demand from service requesters to the service provider to behave in a particular way; capabilities are the service providers promise of behavior. Within the WS-Policy framework, the WS-SecurityPolicy [3] specification can be used to express security requirements and security capabilities in the form of policies. For example, the use of a specific protocol to protect message integrity is a requirement that a service can impose on requesters. On the other hand, the adoption of a particular privacy policy when manipulating data of a requestor is a service capability.

Policy matching in WS-Policy works at a syntactic level: it offers a domain independent mechanism to find alternatives that are compatible to two policies. Two alternatives are compatible if, for each assertion in one alternative, there is a compatible assertion in the other one. Compatibility of assertions is defined exclusively according to the equivalence of their *qnames*, without any further mechanism involving either their structure or their content.

For this reason, several works in the literature [10–13] have been trying to enhance WS-Policy with semantic annotations. In [10], WS-Policy assertions refer to policies expressed in OWL: however that work is not focused on policy matching, but on modeling policies as a set of rules, which have to be evaluated using an external rule-based system, requiring reasoning beyond OWL. In [11] policies represented in WS-Policy are enhanced with semantics by using a combination of OWL and SWRL based rules to capture domain concepts and rules. In [12] a lightweight approach to specify semantic annotations in WS-Policy is presented: it combines the syntactic matching with the semantic matching capability provided by OWL.

Our work, as described in next sections, carefully extends WS-Policy by referencing concepts from a security ontology directly in the policy assertions, thus maintaining backward compatibility with existing tools.

3. System architecture

In this section we discuss the overall architecture of the semantic security policy matching framework. The basic architecture of the system, depicted in Fig. 1, mainly follows that of classic SOA: the requestor, the provider and the registry keep playing their original role in the publish-find-bind cycle. The novelty is the presence of the *Matchmaker* and the *Reasoner* entities (depicted in red), and that of the policy-related information flows (steps 4 through 7).

It is a matter of fact that most of cloud services offered by commercial providers are REST-based. Some providers (Amazon, to cite one) still offer a subset of their services through the SOAP interface, thus providing the relevant WSDL service representation. The architecture depicted in Fig. 1 is compatible with both the REST and the SOAP approach, in the sense that it is capable of performing the security match of services no matter how they have been advertised in the Service Registry and independently of which interface they are going to be finally accessed through. The