Specification of Access Control and Certification Policies for Semantic Web Services

Sudhir Agarwal[1] and Barbara Sprick[2]

[1] Institute of Applied Informatics and Formal Description Methods (AIFB),
University of Karlsruhe (TH), Germany
agarwal@aifb.uni-karlsruhe.de

[2] Department of Computer Science and Automation,
Indian Institute of Science, Bangalore, India
sprick@csa.iisc.ernet.in

Abstract. Web service providers specify access control policies to restrict access to their Web services. It turned out, that since the Web is an open, distributed and dynamic environment, in which a central controlling instance cannot be assumed, capability based access control is most suitable for this purpose. However, since practically every participant can certify capabilities defined in his/her own terminology, determining the semantics of certified capabilities and the trustworthiness of certification authorities are two major challenges in such a setting. In this paper, we show, (1) how certification authorities and their certification policies can be modeled semantically (2) how Web service providers can specify and check the consistency of their access control policies and (3) how end users can check automatically, whether they have access to a Web service.

1 Introduction

Semantic Web services promise dynamic business that can be offered and carried out in the Web. In such an open environment, access control plays an important role. Roughly, access control means that the users must fulfill certain conditions in order to gain access to certain functionality. Access control is not only important from the security point of view, but also from a legal point of view. In some cases, even if a Web service provider does not wish to restrict access to his Web service, he may be forced by law to do so. In our previous works [1,2], we have identified that capability based access control is much more suitable for Semantic Web services than identity based access control methods. In such systems, users prove their legitimacy for access by showing an appropriate set of credentials stating their capabilities. A Web service provider can verify, whether the shown set of proven credentials satisfy all the required constraints. In case, it does not, a Web service provider will not allow the user to access his Web service. However, there are still several open issues that need to be resolved in capability based access control in open environments. We identify three major roles a participant can play in such a setting, namely *user*, *web service provider* and *certification authority.*

K. Bauknecht et al. (Eds.): EC-Web 2005, LNCS 3590, pp. 348–357, 2005.

Users. Users want to access Web services. In case of restricted access they must prove their eligibility, e.g. by showing appropriate certificates (in a capability based setting). In some cases a user wishes to automatically infer whether he/she can fulfil the conditions imposed by the access control policy of a Web service. In another scenario, a user may compose some Web services that may belong to different administrative domains. He then wishes to know the access control requirements of the composed system. To support such use cases, the access control policies as well as the credentials have to be specified formally.

Web Service Providers. Web service providers want to restrict access to their services to only eligible users. For this, they specify and enforce access control policies. If these are specified in terms of capabilities that need to be proven by certificates, then the following questions need to be addressed:

- What is the meaning of the terminology used by CAs in their certificates?
- What is the certification policy of the CA that has issued the certificate?
- Which credentials of which certification authorities shall be trusted? On what grounds are the CAs authorized to certify the respective property?
- Are the credentials still valid or have they been revoked?
- Considering the certification policies of the CAs, is the specified ACP consistent with certain conditions and laws?
- Is the ACP satisfiable? That is, is it possible for any user at all to fulfill all the conditions and thus gain access to the Web service?

Certification Authorities. certify users certain properties by issuing certificates. Each certification authority defines its own terminology that it uses in its certificates, for example, the names of certifiable properties and the relation between certifiable properties.

Currently, certification authorities specify their certification policies explicitly in documents that are readable only for humans (see e.g. [3] for an extensive list of certification authorities). These documents are meant to be read by the service providers before they define the access control policies for their Web services. In this paper we show how certification authorities can specify their certification policies in a machine readable form and how they can publish the policy as well as their certification context, e.g. which properties have been certified to them. This approach has several advantages: When specifying their access control policies, Web service providers can use these certification policies to automatically check, whether the specified ACP fulfills certain (legal as well as self imposed) requirements. When verifying the access eligibility of a particular user on the grounds of a presented set of credentials, the Web service provider can make use of the published certification context to check the validity of the presented credential chains. Though revocation of certificates has been addressed in several papers, it is still a controversial issue (e.g. in the context of delegation) and important real life applications such as *tls* or *ssl* simply ignore the possibility of revocation. Our approach can support the CAs in the implementation and enforcement of revocation of certificates.

We begin with a description of an example scenario that serves as a running example throughout the paper in section 2. In section 3, we introduce a simple and novel approach, how a certification authority can specify explicitly and with machine understandable semantics which terminology it uses in the certificates that it issues and which relationships exist among the certified properties. In section 4, we show how Web service providers can specify the access control policies of their Web services based on their knowledge about certification authorities. We conclude in section 7 after discussing some related work in section 6.

2 Scenario

We now describe an example scenario that will be used throughout the paper as a running example. We consider the following organizational entities: (1) Outdoor Shop (*OutShop*), (2) Wildlife foundation (*WLF*), (3) Forest Department (*FD*), (4) Local Trekking Club (*LTC*).

The Outdoor Shop *OutShop* acts as Web service that maintains a list of approved trekking guides. To register as a guide, a user must be above 25 years old, have good trekking experience as well as knowledge in first aid. *OutShop* trusts the Wildlife foundation *WLF* and the Forest Department *FD* as well as their delegates to certify guidance experience.

The Wildlife Foundation *WLF* issues certificates about guidance experience *app_guide*. In its certification policy it does not relate this properties to any other properties. Furthermore, it allows the local Trekking Club *LTC* to act on its behalf and issue guidance experience certificates.

The Forest Department *FD*, too, issues certificates about guidance experience. However, it issues such certificates only to people who are above 25 years old and have knowledge in first aid (certified e.g. by the Red Cross). This restriction is specified in *FD*'s certification policy.

3 Specification of Certification Policies

In the Web practically every participant can act as *Certification Authority* (CA) autonomously and independent of other CAs. It is therefore unrealistic to assume that there is a global vocabulary of properties that every CA uses (in our example scenario for instance, the meaning of `app_guide` certified by the Forest Department is different from the meaning of `app_guide` certified by the Wildlife Foundation). Further, there may exist logical relationships among certification authorities (e.g. delegation) and among the properties they certify (e.g. inclusion/exclusion of other properties). For example the certification of the property `app_guide` is delegated from *WLF* to the local trekking club *LTC*, and the certification of `app_guide` by the Forest Department implies, that the grantee is above 25 years old and is experienced in first aid.

A signed set of properties that a certification authority certifies together with their relationships and dependencies with other properties (possibly certified by other CAs) is called a *certification policy* of the certification authority. A CA can specify its certification policy and propagandize its trustworthiness in the semantic Web with machine understandable semantics. To do so, the CA specifies

- its certification context in terms of certified properties. This ground can help a Web service provider to decide about the trustworthiness of the certification authority.
- the terminology it uses in the certificates that it issues. These certificates are referenced by a Web service provider in the specification of an access control policy.
- relationships among the properties, that it certifies and any axioms about them.

A certification authority is associated with a set of properties that it possesses and a set of properties that it certifies. In our example, the certification authority *LTC* possesses the property that it is delegated to certifiy property *app_guide* to users. This property is certified by the *WLF*. In order to talk about such properties and certification authorities more formally, we model concepts $\texttt{Property} \sqsubseteq \top$ and $\texttt{CA} \sqsubseteq \top \sqcap \exists \texttt{possesses.Property} \sqcap \exists \texttt{certifies.Property}$.

Delegation. Logical relationships between various CAs and the properties they certify can be specified through axioms, in particular class hierarchies over relevant instances of `Property`. Note, that the concrete properties used in an axiom by a certification authority C do not necessarily need to be the properties that are certified by C. Subclass relationship between two classes of different certification authorities can be used for specifying delegation structures. Consider for example, certification authorities C_1 and C_2 that certify properties P_1 and P_2, respectively. By defining the axiom $P_1 \sqsubseteq P_2$, the certification authority C_2 states, that anyone possessing the property P_1 also possesses the property P_2. In other words, the certification authority C_2 delegates the certification of the property P_2 to the certification authority C_1.

Enforcement. We propose to use two kinds of certificates for the enforcement of certification policies, namely *delegation certificates* and *property certificates.* Delegation certificates are meant for realizing the delegation. A delegation certificate is issued by a certification authority to another certification authority to allow the latter to issue certificates (delegation or property) about a property that is defined by the former. This certificate will be signed by the issuer and contain the public key of the recipient. A property certificate is a certificate that is issued by a certification authority to an agent certifying that this agent has a certain property. A property certificate is signed by the certification authority and contains the public key of the recipient.

3.1 Certification Policy - Example

Recalling our running example, we consider the certification authorities, namely the *WLF*, *FD*, *LTC*, *GOV* and *REDCROSS* that we model as instances of the concept `CA` as `CA(WLF)`, `CA(LTC)`, `CA(FD)`, `CA(GOV)`, `CA(REDCROSS)`.

WLF defines and certifies the property "appguide" to approved guides, which can be modeled with the following axioms.

$$\texttt{Property(WLF.org;\#appguide)}$$
$$\texttt{certifies(WLF,WLF.org;\#appguide)}$$

Further, *WLF* defines

$$\texttt{LTC.org;\#appguide} \sqsubseteq \texttt{WLF.org;\#appguide} \tag{1}$$

to specify the delegation structure which mean that the guides approved by *LTC* are also approved guides from the point of view of *WLF*. By specifying such an axiom, *WLF* delegates the certification of the property "`WLF.org;#appguide`" to *LTC*.

LTC and *FD* also issues certificates to approved guides. Government *GOV* certifies the property `state.gov;#above25` to people who are above 25 years of age. Similarly, *REDCROSS* issues certifies the property `RedCross.org;#firstaid` to persons who have experience in first aid.

$$\texttt{certifies}(\texttt{LTC}, \texttt{LTC.org;\#appguide})$$
$$\texttt{Property}(\texttt{FD.org;\#appguide})$$
$$\texttt{certifies}(\texttt{FD}, \texttt{FD.org;\#appguide})$$
$$\texttt{certifies}(\texttt{GOV}, \texttt{state.gov;\#above25})$$
$$\texttt{certifies}(\texttt{REDCROSS}, \texttt{RedCross.org;\#firstaid})$$

Though on first sight the properties `WLF.org;#appguide` and `FD.org;#appguidefd` seem to be equivalent because of their names, they are quite different in their certification policies: The Forest Department certifies this propety only to guides who are at least 25 years old and who are knowledgable in first aid, certified by the government. This restriction can be expressed in the certification policy by stating the following axiom:

$$\texttt{FD.org;\#appguide} \sqsubseteq \texttt{state.gov;\#above25} \sqcap \texttt{RedCross.org;\#firstaid} \tag{2}$$

Note the difference between the two axioms (1) and (2): While in axiom (1) the conclusion of the axiom lies in the namespace of specifying CA, it is the assumption of axiom (2). In this sense, axiom (1) has more the character of a definition (of delegation) and may substitute a delegation credential. Axiom (2) is more a promise or an actual certification policy which says, that the CA *FD* "promises" to check the age and the knowledge about first aid before certifying a user that he is an approved guide. In case there is no axiom in the government's namespace delegating the certification of the property *age above 25* and in the namespace of the Red Cross delegating the certification of the property knowledgeable in First Aid, a potential Web service provider now has to decide whether to trust this policy or not.

4 Specification of Access Control Policies

We now turn our attention to Web service providers and consider the problem how they can restrict access to their services. In a capability based setting, access is granted or denied on the basis of certified properties. For a correct specification of the access control policy, a Web service provider faces the following two problems: (1) he must understand the meaning of credentials and certified properties and (2) he must trust the issuers of credentials. A Web service provider can understand the meaning of the credentials issued by a certification authority from the description of the properties that the certification authority certifies. On the basis of the description of the properties that a certification authority possesses, a Web service provider can build his trust in the certification authority.

We now show how a Web service provider can make use of certification policies as introduced in section 3 while (1) *specifying* the access control policy for his service and (2) *verifying* the eligibility for access of a particular user.

Specifying Access Control Policies. An *access control policy* for a Web service w is a set of *authorization terms* (p, w, f). Each authorization term has the intuitive meaning that a user being able to prove property p is granted access to functionality f of Web service w. The tuple $\langle w, f\rangle$ is called an *interface*, the set of all interfaces is denoted by I. We define the *expansion* $exp(p, w, f)$ of an authorization term to be the set $\{(s, w, f)|$ subject s can prove to have property $p\}$ and the expansion $exp(\Pi)$ of a policy Π to be the union of the expansions of elements of Π. An authorization term can be defined with DL axioms as follows:

$$\begin{aligned}\texttt{AuthorizationTerm} \sqsubseteq\ & \top \sqcap \exists \texttt{subject.ca-Property} \sqcap \\ & \exists \texttt{object.WebService} \sqcap \\ & \exists \texttt{authorization.WebServiceFunctionality}\end{aligned}$$

An approach for specifying and automatically composing access control policies using a policy algebra [4,5] has been proposed in [1,2]. The algebra allows to specify and compute access control policies of composite Web services from those of its component Web services.

Verifying Eligibility of a User. When a user requests access by showing his set of certificates the Web service provider must be able to verify the user's eligibility in order to decide whether to grant or to deny access to the user. To verify the eligibility of a user, he checks for each access requirement, whether the shown set of credentials (plus possibly published certification policies and delegation credentials published by relevant CAs) contain a valid certificate chain from some trusted CA to the required property. However, in this process, the service provider also needs to consider the possible revocation of one or more certificates shown by the user. This could either be a certificate directly issued to the user or a certificate in the certificate chain issued to one of the involved CAs. If the CAs publish their certification policies including the delegation credentials issued to them in machine readable form on the web, it might still be possible to automatically find a valid chain that proves the users' eligibility, e.g. via newly stated relationships or other published delegation certificates.

4.1 Access Control Policy - Example

Let us now consider our running example again to illustrate how a Web service provider can specify an access control policy using the knowledge he gains from the certification policies of the certification authorities.

The outdoor shop *OutShop* offers a registration service for approved trekking guides. For the registration they have the following access condition: Each trekking guide must be approved by either the Forest department *FD* or by the wildlife foundation *WLF*, must at least 25 years old and must be knowledgeable in first aid. This leads to the following access control policy:

$$ACP(P) := \{(WLF, \texttt{OS.edu}, \textit{trecking guide}), (FD, \texttt{OS.edu}, \ \textit{trecking guide})\}$$

where WLF and FS are defined as follows:

$$WLF \equiv \texttt{WLF.org;\#appguide} \sqcap \texttt{state.gov;\#above25} \sqcap \texttt{RedCross.org;\#firstaid}$$
$$FD \equiv \texttt{FD.org;\#appguide} \sqcap \texttt{state.gov;\#above25} \sqcap \texttt{RedCross.org;\#firstaid}$$

From the certification policy, the Web service provider can infer that whoever has the property FD.org;#appguide also has the properties state.gov;#above25 and RedCross.org;#firstaid. At the time of specification of the policy, it allows him to relax the policy and thus reduce the number of certificates, a potential user has to present:

$$ACP(P) := \{(WLF, \texttt{OS.edu}, \textit{trecking guide}), (FD, \texttt{OS.edu}, \ \textit{trecking guide})\}$$

with FS being defined as

$$FD' \equiv \texttt{FD.org;\#appguide}.$$

At the time of verification it allows the Web service provider to verify, that a user, who has presented "only" a valid chain for FD.org;#appguide is still eligible for the registration.

5 Users

We now turn our attention to the third logical role, namely *users*. Users are mainly interested in accessing Web services. In case of secure semantic Web services, which is our main concern in this paper, any Web service discovery component must consider the user's certificates as well as the access control policies of Web services. Syntactical certificates description schemas, such as X509, KeyNote or SPKI/SDSI certificates make it difficult for a client side discovery component to perform matching based on functional as well as security aspects, because such a discovery component can not know the meaning of the certificates that the user has and hence can not know which properties has been certified to the user.

In our setting, an end user possesses a set of certificates meta-data about each certificate. The meta-data contains information about the properties that a certificate actually certifies and hence describes the semantics of the certificate. Note, that in many cases, a certificate certifies more than one property. The end users (1) have their goals in mind and want to discover and compose Web services, (2) want to access Web services that offer required functionality, (3) have certain properties certified to them and can use the certificates to prove their eligibility if access to a Web service is restricted.

5.1 Example

Consider an end user, who is an experienced wildlife and trekking guide and holds a certificate certifying him the property `LTC.org;#appguide`. The user wishes to register himself as an approved trekking guide in the Outdoor Shop *OutShop*.

In our example, the set of Web service descriptions that our end user obtains from an appropriate discovery component will contain the description of the Web service *OutShop*, since the matching software can infer from the certification policy of the wildlife foundation WLF that `LTC.org;#appguide` is subset of `WLF.org;#appguide` and hence a user possessing the property`LTC.org;` #appguideltc has access to the Web service *OutShop*

Now, our end user wants to register as a approved trekking guide. He finds out, that this functionality is accessible only for approved guides from the forest department FD and from the Wildlife foundation WLF, that are above 25 years old and are knowledgeable in first aid. By looking at the certification policies of WLF and FD he finds out automatically, that WLF has delegated the certification of property *app_guide* to the local trekking club TLC, which means, that it will be enough to hold the certificate `LTC.org;#appguide` rather then the certificate `WLF.org;#appguide`.

6 Related Work

In the area of service oriented computing there are already several approaches for declaratively modeling the user's objectives; mainly in terms of policies. On the one hand, there are XML-based approaches, like WS-Security, XACML, EPAL, etc. These approaches allow to model constraints about domain-specific attributes of a service. XACML has been approved by OASIS and that promises to standardize policy management and access decisions. However, XACML focusses more on technical issues and addresses how the access control can be enforced. EPAL and XACML specifications greatly overlap and do very similar things in slightly different ways. As a consequence, the user has to learn the different approaches and work with different policy tools. Furthermore, it is not possible to specify a policy that combines privacy and communication security concerns such as: send sensitive content over secured lines only. WS-Policy introduces a logic framework that allows domain-specific policy assertions to be plugged in. Nevertheless, the supported assertions are very simplistic in nature and still require the respective native policy interpreters. The major disadvantage of XML is that the semantics is contained implicity in the expressions. Meaning arises only from the shared understanding derived from human consensus. This leads to extra manual work for software engineers and could easily result in fragmentation.

Security-related ontologies to markup DAML-S [6] elements such as input and output parameters with respect to their security characteristics, such as encryption and digital signatures have been developed in [7,8]. [9] gives an short introduction to Rei, case studies, use cases and open issues. However, the mechanism described in the paper requires clients to send their privacy policies and permissions to a Web service provider, which is not always wishful. Nevertheless, the authors identify the enforcement problem as an open issue. [10] introduces an enforcement architecture based on a policy engine and the policy enforcement mechanism for pervasive environments. The policy engine reasons over policies described in Rei and uses Prolog for its reasoning engine. [11] discusses the policy language Rei in more detail. However, neither [10] nor [11] provide Rei's mapping to Prolog. So, it is not clear what the policy engine acutally does. Since Rei requires a special reasoner, it is not clear, what is the added value of Rei as compared to XML based approaches except that it is more expressive.

Our work is complementary to the existing approaches as it also addresses the need of machine understandable specification of certification policies that are specified by the certification authorities. It also presents how Web services providers can use such semantically-rich specifications for defining their access control policies. Consequently, our paper covers a broader spectrum and also shows the added value of specifications with formal semantics within the context of access control. We also address the issue of enforcement which can not be ignored while dealing with security related aspects. Finally, our approach is completely based on description logics which is the formalism behind the W3C standard OWL (Web Ontology Language). Consequently, we do not require the users and providers of semantic Web services to install a special reasoner.

7 Conclusion

In a capability based access control system a Web service provider specifies the access requirements for his service in terms of required properties. For gaining access, users have to prove that they satisfy these properties. To do so, they need to present certificate chains that prove a delegation chain from a trusted certification authority to the required property.

This approach faces several problems: revocation of certificates is still a controversial issue since the semantics is not fully clear in presence of delegation. The second problem is that one cannot always assume the existence of a "trusted" root CA, e.g. Verisign, as in principle everyone can act as CA. Potential CAs therefore publish their (signed) certification policies in order to establish trust in them. Thirdly, while composing Web services and access control policies automatically from those of its components, a Web service provider needs to check whether governmental- and self imposed laws are still met by those CAs trusted by the components. The Web service provider is then able to automatically check whether the trust structure of the component services is compatible with his trust structure. Our approach shows how certification policies can be specified in the semantic Web in a machine understandable way, which makes them suitable for automatic verification of the compatibility between access control policies and governmental or self imposed laws. Further, delegation structures of CAs can be made explicit in our approach, which allows CAs to handle revocation of (delegation) certificates in an online manner.

References

1. Agarwal, S., Sprick, B.: Access control for semantic web services. In: 1st International Conference on Web Services. (2004)
2. Agarwal, S., Sprick, B., Wortmann, S.: Credential based access control for semantic web services. In: AAAI Spring Symposium 2004 - Semantic Web Services. (2004)
3. Stefan Kelm, S.S.C.: The pki page – extensive list of certification authorities. http://www.pki-page.org/ (2004)
4. Biskup, J., Wortmann, S.: Towards a credential-based implementation of compound access control policies. Technical report, University of Dortmund (2003) http://ls6-www.cs.uni-dortmund.de/issi/publications.
5. Bonatti, P., de Capitani di Vimercati, S., Samarati, P.: An algebra for composing access control policies. ACM Transactions on Information and System Security (TISSEC) **5** (2002) 1–35
6. Ankolekar, A., Burstein, M.H., Hobbs, J.R., Lassila, O., Martin, D., McDermott, D.V., McIlraith, S.A., Narayanan, S., Paolucci, M., Payne, T.R., Sycara, K.: DAML-S: Web Service Description for the Semantic Web. In: ISWC2002: Ist International Semantic Web Conference, Sardinia, Italy. Lecture Notes in Computer Science, Springer (2002) 348–363
7. Denker, G., Kagal, L., Finin, T., Sycara, K., Paolucci, M.: Security for daml web services: Annotation and matchmaking. In: Second International Semantic Web Conference. Volume 2870 of Lecture Notes in Computer Science., Springer (2003)
8. Kagal, L., Paolucci, M., Srinivasan, N., Denker, G., Finin, T., and, K.S.: Authorization and privacy for semantic web services. In: Proc. of AAAI Spring Symposium on Semantic Web Services. (2004)
9. Kagal, L., Finin, T., Joshi, A.: Declarative Policies for Describing Web Service Capabilities and Constraints. In: W3C Workshop on Constraints and Capabilities for Web Services, Oracle Conference Center, Redwood Shores, CA, USA, W3C (2004)
10. Patwardhan, A., Korolev, V., Kagal, L., Joshi, A.: Enforcing Policies in Pervasive Environments. In: International Conference on Mobile and Ubiquitous Systems: Networking and Services, Cambridge, MA, IEEE (2004)
11. Kagal, L., Finin, T., Joshi, A.: A Policy Language for A Pervasive Computing Environment. In: IEEE 4th International Workshop on Policies for Distributed Systems and Networks. (2003)