Access Control for Semantic Web Services

Sudhir Agarwal
Institute of Applied Informatics and
Formal Description Methods (AIFB),
University of Karlsruhe, Germany
agarwal@aifb.uni-karlsruhe.de

Barbara Sprick
Information Systems and Security,
Department of Computer Science,
University of Dortmund, Germany
barbara.sprick@udo.edu

Abstract

In this paper, we present an approach to enable access control for semantic web services. Our approach builds on the idea of autonomous granting of access rights, decision making based on independent trust structures and respects privacy requirements of the users. Our framework allows the specification and computation of complex access control policies in a manageable and efficient way. Therefore, our approach is useful not only in web services based applications (typically client-server architecture) but also in peer to peer and agent based applications.

1. Introduction

With the advent of the semantic Web [2, 12, 14], Web services have gained even more importance [6]. semantic Web techniques, especially ontologies, allow to describe Web services with machine understandable semantics, thus enabling new features like automatic composition, simulation and discovery of Web services [6].

Because of the vast heterogeneity of the available information, information providers and users, security becomes extremely important. Security related aspects are mostly classified in three categories, namely confidentiality, integrity and availability [3, 16, 9]. Access control, which means the users must fulfill certain conditions in order to access certain functionality plays an important role in all three fields.

We identify the following **desiderata** for access control mechanisms for semantic Web services:

1. Access control should not require central control and should allow providers to specify access control policies autonomously.
2. Access control should be based on capabilities rather than identities.
3. The framework should allow end users to check and prove her eligibility for a Web service.
4. The framework should support the specification and computation of complex (composite) access control policies.
5. The specification of access control policies of composite Web services should be immune to the changes in the access control policies of its component Web services.
6. The framework should allow to identify the required partial behaviour of a composite Web service and to compute and integrate the access control requirements of the partial behaviour of a component Web service.
7. The framework should be able to deal with contracts that have impact on the access control policy of a component Web service.
8. The framework should be able to deal with the capabilities that are certified to the requester on the fly, that is, during the execution of a composite Web service.
9. The framework should be able to specify consumable credentials.
10. The framework should be able to specify the validity and reason about the execution time of a Web services.
11. The framework should make sure that information delivered by a component Web service as well as credentials shown by requesters are not misused by the composite Web service.

In section 2, we give a short introduction to policy algebra introduced in [5]. In section 3, we present our main contribution by introducing an approach for specification and computation of access control policies. In section 4, we show how our approach can be integrated with DAML-S and implemented with SPKI/SDSI. In section 5, we discuss some related work and conclude.

2. Introduction to Policy-Algebra

We now introduce an algebra for composing access control policies similar to the one described in [5].

An *access control policy* for a Web service o 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 $\{(s, w, f)| \quad \text{subject } s \text{ can prove to have property } p\}$ and the expansion $exp(\Pi)$ of a policy Π to the be union of the expansions of elements of Π.

Policy expressions are syntactically built from policy identifiers and algebra operators as follows:
E ::= $\mathbf{id} \mid E{+}E \mid E \,\&\, E \mid E{-}E \mid E^C \mid T(E) \mid (E)$
T ::= $\tau\mathbf{id}.E$
Here, **id** is the token type of policy identifiers, E is the nonterminal describing *policy expressions*, $C \in I$ is an interface restriction on policies, and T is a *template*, that represents partially specified policies. Note, that the templates are not policy expressions, only templates with actual parameters are.

Policy identifiers are interpreted by an environment that maps policy identifiers to policies. The semantics of the policy algebra is a function e that maps each policy expression to an expanded policy, inductively extending an environment by using the pertinent interpretation of the operators, and each template onto a function over policies. The operators addition (+), the conjunction (&) and subtraction (−) are interpreted as set-theoretic union, intersection and difference on the expanded policies, respectively. The scoping restriction "C" restricts a policy to the interfaces $\langle w, f \rangle \in C$.

3. Computation of Composite ACPs

We distinguish between atomic Web services and composite Web services. Composite Web services are composed from component Web services by operations *Sequence (;), Choice (+), Parallel (||), Iteration (∗)*. The provider of an atomic Web service can specify the access control policy of her service autonomously and independently as a set of authorization terms of form (p, w, f). The provider of a composite Web service w computes the access control policy of w from those of its components (cf. desideratum 4). For the computation, the access control policies of the desired functionalities of the component Web services, the type of their composition and possible contracts between the provider of the composite Web service and the component Web services need to be taken into account.

Support for partial functionalities Often, a composite Web service needs only a partial functionality of a component (cf. desideratum 6). Each functionality of a Web service is specified as an interface $\langle o, a \rangle$. We use the scoping operator for restricting the access control policies of a Web service to the access control policy of the *desired functionality*.

Consider, for example, a component Web service w with functionalities $\{f_1, f_2\}$. The access control policy $\Pi(w)$ will contain authorizations for interfaces $\langle w, f_1 \rangle$ and $\langle w, f_2 \rangle$. If composite Web service w' only requires functionality f_1 of service w, the access control policy of service w' should contain authorization terms only for interface $\langle w, f_1 \rangle$. The restriction of access control policy $\Pi(w)$ to the interface $\langle w, f_1 \rangle$ is then defined as $\Pi(w)^{\{\langle w, f_1 \rangle\}}$.

Support for independently specified access control policies Desideratum 1 implies that a provider of a composite Web service does not know at design time how the access control policy of a component Web service will look like at the instantiation time. We support this by using the template operator which allows to specify that the access control policy of a composite service contains the access control policy of a component service without actually inserting the component access control policy at the design time. The templates will be instantiated when the access control policy needs to be computed at the time of instantiation of the composite Web service.

Consider for example a Web service w that is specified as a sequence of component Web services w_1 and w_2. Obviously, the access control policies of the Web services w_1 and w_2 must be contained in the specification of the access control policy of the Web service w. However, since the providers of Web services w_1 and w_2 can specify and modify access control policies of their respective Web service without notifying the provider of w, it is more appropriate if the provider of the composite Web service w uses template $\tau.X_1, X_2.(X_1 \& X_2)$.

Support for contracts Possible contracts between Web services (cf. desideratum 7) may affect the access control policies of the Web services. In such a case, the provider of the composite Web service has to replace a part of the services' access control policy with a policy the providers agreed upon in a contract.

Consider for example a contract that specifies that users who use component Web service w_1 via composite Web service w may give the bank account details instead of showing a credit card. The access control policy of Web service w is then calculated by the access control policies of the other involved Web services and $(\Pi(w_1) - \Pi(\mathit{CreditCard}))\&(\Pi(\mathit{CreditCard}) + \Pi(\mathit{BankAccount}))$.

Computation of access control policies We now show the computation of a composite access control policy from component access control policies (cf. desideratum 4). We consider the control constructs *Sequence (;), Choice (+), Parallel (||), Iteration (*)*.

The composite access control policy is computed in two steps: In the first step, we adapt the interfaces from component services to interfaces of the composite service. In the second step, we actually compute the composite access control policies.

Let $I(w_1)$ and $I(w_2)$ denote the respective interfaces for Web services w_1 and w_2. The set of interfaces for the composite service $w = w_1; w_2$ is defined as

$$I(w) \quad := \{\langle w, f_1 f_2\rangle | \quad \langle w_1, f_1\rangle \in I(w_1) \text{ and } \langle w_2, f_2\rangle \in I(w_2)\}$$

The adaption $\Pi'(w_1)$ and $\Pi'(w_2)$ of policies $\Pi(w_1)$ and $\Pi(w_2)$, respectively, is the defined as follows:

$$\begin{aligned}\Pi'(w_1) \quad &:= \{(p, w, f_1 f_2) | \quad \langle w, f_1 f_2\rangle \in I(w) \text{ and } (p, w_1, f_1) \in \Pi(w_1)\}\\ \Pi'(w_2) \quad &:= \{(p, w, f_1 f_2) | \quad \langle w, f_1 f_2\rangle \in I(w) \text{ and } (p, w_2, f_2) \in \Pi(w_2)\}\end{aligned}$$

The interface and policy adaption for control constructs Choice (+) and Parallel (||) is analogous. The computation of the composite access control policy for each control construct is then defined as follows:[1]

$\Pi(w_1; w_2)$	$\tau.X_1, X_2.(X_1\&X_2)(\Pi'(w_1), \Pi'(w_2))$
$\Pi(w_1 + w_2)$	$\tau.X_1, X_2.(X_1 + X_2)(\Pi'(w_1), \Pi'(w_2))$
$\Pi(w_1 \| w_2)$	$\tau.X_1, X_2.(X_1\&X_2)(\Pi'(w_1), \Pi'(w_2))$
$\Pi(w^*)$	$\tau.X_1.(X_1)(\Pi(w))$

In case, Web services are composed sequentially or in parallel, the access control policy of the composite services is simply the conjunction of the component services, the requester of the composite Web service needs to fulfill the access control policies of both component services. In case, the Web service is composed as a choice between two component Web services, the requester needs to fulfill the access control policy of at least one the of the component services. In case, the composite Web service is an iteration of the component Web service, the requester needs to fulfill the access control policy of the component service.

1 Note that we assume that the component Web services w_i are already relevant parts of the Web services $\bar{w_i}$ already identified by scoping, contracts etc.

Note, that we assumed a capability based access control system where capabilities do not get revoked by using them.

4. Implementation

Desideratum 2 states that access control should be based on capabilities rather than on identities. An authorization term (p, o, a) specifies, that a user must be able to prove property p to access interface $< o, a >$.

We implement the policy algebra introduced in section 2 with an extension of SPKI/SDSI as proposed in [4]. SPKI/SDSI is a credential based public key infrastructure resulted by merging SDSI (Simple Distributed Security Infrastructure) and SPKI (Simple Public Key Infrastructure). The main advantage of SPKI/SDSI compared to other credential based systems is that it does not require central control and allows users, e.g., Web service providers to specify their own trust structures independent of each other. SPKI/SDSI supports two kinds of credentials, namely *name certificates* to bind principals to names and *authorization certificates* to bind authorizations to names. Besides name certificates and authorization certificates, SPKI/SDSI also provides access control lists (ACL) for specifying access control policies for some interface [1, 10, 11, 15].

The explicit enumeration of wanted authorization terms $(s_1, o, a), \ldots, (s_n, o, a)$ for a policy identifier P_i is implemented as follows. A trusted assigner, K_{asn_i}, defines an appropriate local name $K_{asn_i} N_i$ by issuing a name certificate to each Principal K_{s_l}, denoting an agent s_l. An algebraic policy expression P is implemented as follows. The controller declares an authorization certificate with `delegation` set to `false` and `subject` is an algebra expression. We assume that credentials are always valid except that they are in some revocation list. This means that we do not consider one time credentials as well as credentials with limited validity. The algebra expression is the policy expression P' that is the policy expression P where (1) each occurrence of selection is replaced by conjunction and each constraint is replaced by an appropriate local name $K_{asn_j}\mathit{constraint}$ and (2) each policy identifier P_i is replaced by an appropriate local name $K_{asn_i} N_i$.

Then the trusted assigners define their local names by issuing name certificates of the simplest form (i.e. local name are explicitly defined by principals) to each principal that denotes an agent who in case (1) satisfies the demanded constraint and who in case (2) is element of the interpreted respective policy identifier. For further details, especially about the semantics of a policy expression P, we refer to [4].

4.1. Access Control with DAML-S

We view the access control policy of a Web service as a condition that a user has to fulfill to get access to the Web service. We model a concept `ACCondition` as subclass of `Condition`. `ACCondition` has properties `acp` of type `ACPolicy` and `inputParameter`. A precondition of type `ACCondition` means that the input parameter referred to by the property `inputParameter` should prove the satisfiability of the access control policy referred to by the property `acp`.

Concept `ACPolicy` with property `authorizationTerm` represent an access control policyand the set of authorization terms, the access control policy consists of. The concept `AuthorizationTerm` with properties `property`, `object` and `authorization` represents a triple of the form (p, o, a). Setting the range of the property `property` to `Capabilities` allows to specify users of a Web service based on their properties (e.g. public key) and not on their identity. The `object` is the Web service itself. Hence, we set the range of `object` to `Process`. We set the range of `authorization` to `ConditionalOutput` since a conditional output corresponds to a functionality offered by a Web service. The policy algebra operators and constructs can be modeled rather straightforward.

5. Related Work and Conclusion

In [8], authors give several security-related ontologies that are designed to represent well-known security concepts. In [13], authors introduce a policy language that allows policies to be described in terms deontic concepts and models speech acts. In [7], authors propose adding privacy and authentication annotations, for example, cryptographic type, to input and output parameters to aid in selection of semantic web services.

In this paper, we identified some important access control related requirements and presented an approach for specifying access control policies for semantic web services. We showed, how access control policies of composite web services can be computed from the structure of the composite web service and the access control policies of its component web services. We presented how such specifications can be implemented by using SPKI/SDSI and integrated with DAML-S.

6. Acknowledgements

We thank the anonymous reviewers for their helpful remarks and pointers. This work was funded by the DFG-Project "Kompositionale Credential-basierte Zugriffskontroll-Systeme" (BI 311/11-1) and by the BMBF-Project "Internetökonomie".

References

[1] S. Agarwal, B. Sprick, and S. Wortmann. Credential based access control for semantic web services. In *AAAI Spring Symposium - Semantic Web Services*, 2004.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 2001.

[3] M. Bishop. *Computer Security – Art and Science*. Addison Wesley, 2003.

[4] J. Biskup and S. Wortmann. Towards a credential-based implementation of compound access control policies. In *Symposium on Access Control Models and Technologies*, IBM, Yorktown Heights, New York, June 2004.

[5] P. Bonatti, S. de Capitani di Vimercati, and P. Samarati. An algebra for composing access control policies. *ACM Transactions on Information and System Security (TISSEC)*, 5(1):1–35, 2002.

[6] DAML-S Coalition. DAML-S: Web Service Description for the Semantic Web. In *ISWC2002, Sardinia, Italy*, volume 2342 of *LNCS*, pages 348–363. Springer, 2002.

[7] G. Denker, L. Kagal, T. Finin, M. Paolucci, N. Srinivasan, and K. Sycara. Sowl: A security infrastructure for owl-s - an approach to confidentiality and integrity. In *AAAI Spring Symposium - Semantic Web Services*, 2004.

[8] G. Denker, L. Kagal, T. Finin, K. Sycara, and M. Paolucci. Security for daml web services: Annotation and matchmaking. In *ISWC2002, Sardinia, Italy*, volume 2342 of *LNCS*. Springer, 2003.

[9] D. E. Denning. *Cryptography and Data Security*. Addison Wesley, 1982.

[10] C. M. Ellison, B. Frantz, B. Lampson, R. L. Rivest, B. M. Thomas, and T. Ylonen. Simple public key certificate. http://world.std.com/˜cme/html/spki.html, 1999.

[11] C. M. Ellison, B. Frantz, B. Lampson, R. L. Rivest, B. M. Thomas, and T. Ylonen. SPKI certificate theory. Internet RFC 2693, 1999.

[12] D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, editors. *Spinning the Semantic Web*. MIT Press, 2002.

[13] L. Kagal, T. Finin, and A. Joshi. A policy based approach to security for the semantic web. In *ISWC2003*, volume 2870 of *LNCS*. Springer, 2003.

[14] P. Patel-Schneider and D. Fensel. Layering the semantic web: Problems and directions. In *ISWC2002, Sardinia, Italy*, volume 2342 of *LNCS*, pages 16–29. Springer.

[15] R. L. Rivest and B. Lampson. SDSI - a simple distributed security infrastructure. http://theory.lcs.mit.edu/˜cis/sdsi.html, 1996.

[16] P. Samarati and S. Capitani di Vimercati. Access control: policies, models, and mechanisms. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design (FOSAD)*, volume 2171 of *LNCS*, pages 137–196. FOSAD 2000, Bertinoro, Italy, Springer Verlag, Berlin, 2001.