PrimeLife Policy Language

Claudio A. Ardagna[2], Laurent Bussard[3], Sabrina De Capitani di Vimercati[2], Gregory Neven[4], Stefano Paraboschi[1], Eros Pedrini[2], Franz-Stefan Preiss[4], Dave Raggett[5], Pierangela Samarati[2], Slim Trabelsi[4], Mario Verdicchio[1]

DIIMM - Università degli Studi di Bergamo, 24044 Dalmine, Italy[1]
DTI - Università degli Studi di Milano, 26013 Crema, Italy[2]
European Microsoft Innovation Center (EMIC), 52072 Aachen, Germany[3]
IBM Zurich Research Center, Zurich, Switzerland[4]
SAP Labs France, Sophia Antipolis, France[4]
W3C/ERCIM [5]

Contact author: Slim Trabelsi (slim.trabelsi@sap.com)

Introduction

The sudden popularity of social networks and web 2.0 applications changed radically the Internet landscape and the users' behavior. Today's young people are the first generation with the ability to distribute information quickly, cheaply and to large groups of people. The amount of personal and private information published and stored in the servers becomes so huge that the traditional concepts of privacy were radically affected. To appease such concerns, enterprises and service providers publish privacy statements that promise fair information practices. Written in natural language or formalized using languages like P3P [1], EPAL [2], XACML [3] etc... they are only promises but not necessarily enforced by technical measures. These problems are amplified if personal data is used not only by the enterprise that collected the data, but also by secondary users such as partner organizations, or government agencies. These flows of data are complex. Threats to data privacy can come from inside (accidental disclosure, insider curiosity and subornation) as well as from the outside (uncontrolled secondary usage) of each organization. Putting customer information online further increases the risk of exposing private and sensitive information to outsiders. In this paper we propose a new policy language handling access control and data usage at the same time. In the context of the European ICT PrimeLife[1] we propose an extension of the eXtensible access control markup language (XACML 3.0) offering one of the most popular standardized policy language. This extension suggests a new obligation handling mechanism taking into account temporal constraints, pre-obligations, conditional obligations, and repeating obligations together with a down-stream usage authorization system defining the access control rules under which personal information collected by an entity can be forwarded to a third party. Moreover, our language is based on the concept of trusted credentials.

[1] http://www.primelife.eu/

PrimeLife Policy Language

As mentioned previously we extend XACML 3.0 with data handling and credential capabilities. We maintain the overall structure of the XACML language, but we introduce a number of new elements to support the advanced features that our language has to offer, and we also modify the schema of a number of existing elements. Our language is intended to be used by the Data Controller[2] to specify the access restrictions to the resources that he offers; by the Data Subject[3] to specify access restrictions to her personal information, and how she wants her information to be treated by the Data Controller afterwards; by the Data Controller to specify how "implicitly" collected personal information (such as IP address, connection time, etc.) will be treated; and by the Data Subject to specify how it wants this implicit information to be treated. The following sections describes how XACML is extended, what are the new elements that we propose, and why is it essential to add such extensions.

a. Rules, policies, and policy sets

As in XACML, the main components of our language are rules, policies, and policy sets. Each rule has an effect, either "Permit" or "Deny", that indicates the consequence when all conditions stated in the rule have been satisfied. Rules are grouped together in policies. When a policy is evaluated, the rule combining algorithm of the policy (as stated in an XML attribute of the policy) defines how the effects of the applicable rules are combined to determine the effect of the policy. The main components of a rule are a target, describing the resource, the subject, and the environment variables for which this rule is applicable; credential requirements, describing the credentials that need to be presented in order to be granted access to the resource; provisional actions, describing which actions have to be performed by the requestor in order to be granted access; a condition, specifying further restrictions on the applicability of the rule beyond those specified in the target and the credential requirements; data handling policies, describing how the information that needs to be revealed to satisfy this rule will be treated afterwards; and data handling preferences, describing how the information contained in the resource that is protected by this rule has to be treated.

b. Obligations

i. Introduction to Obligations

We define an obligation as: "A promise made by a data controller to a data subject in relation to the handling of his/her personal data. The data controller is expected to fulfill the promise by executing and/or preventing a specific action after a particular event, e.g. time, and optionally under certain conditions". Obligations play an important role in daily business. Most companies collect personally identifiable information (personal data) on customers and employ ad-hoc mechanisms to keep track of associated authorizations and obligations. State of the art mechanisms to handle collected personal data in accordance with to a privacy policy are lacking expressiveness and/or support for cross-domain definition of obligations.

[2] A Data Controller is an entity that alone or jointly with others determines the purposes and means of the processing of personal data. The processing of personal data may be carried out by a Data Processor acting on behalf of the Data Controller.

[3] A Data Subject is the person whose personal data are collected, held or processed by the Data Controller.

ii. Why should we define an Obligation Language?

Most of the available policy languages, like XACML [3], EPAL [2], Ponder [4], Rei [5] and PRIME-DHP [1], provide either only a placeholder or very limited obligation capability. Moreover these languages do not provide any concrete model for obligation specification. The work proposed in this paper incorporates some of the prior art and extends it toward more expressiveness, extensibility, and interoperability. We identify four main challenges related to obligations:

- Service providers must avoid committing to obligations that cannot be enforced. For instance, it is not straightforward to delete data when backup copies do exist. Tools to detect inconsistencies are necessary.
- Services should offer a way to take user's preferences[4] into account. Preferences may be expressed by ticking check boxes, by a full policy, or even be provided by a trusted third party. Mechanisms to match user's privacy preferences and service's privacy policies are necessary.
- Services need a way to communicate acceptable obligations to users, to link obligations and personal data, and to enforce obligations.
- Finally, users need a way to evaluate the trustworthiness of service providers, i.e. know whether the obligation will indeed be enforced. This could be achieved by assuming that misbehavior impacts reputation, by audit and certification mechanisms, and/or by relying on trusted computing.

c. Definition of an Obligation Language

An obligation is often defined as Event-Condition-Action [4]

On ***Event*** If ***Condition*** Do ***Action***

For facilitating the comparison of obligations, we consider triggers as events filtered by conditions. In other words, we replace the notions of events and conditions by trigger. The triggers are events related to an obligation. These events result in actions that are executed according to the obligation's requirements. Additionally, in order to simplify obligations management, we specify a validity period for each obligation:

Do ***Action*** when ***Trigger*** (from Start to End)

We use a common language for expressing obligations in data controller's privacy policy, in data subject's privacy preferences, and in sticky policies.

- Data subject's privacy preferences specify "required obligations", i.e. what the data subject requires in terms of obligation to provide a given piece of personal data to a given data controller.
- Data controller's privacy policy specifies "proposed obligations", i.e. what the data controller is willing (and able) to enforce in terms of obligation for a given collected data.

Sticky policy specifies "committed obligations", i.e. the obligations data subject and data controller agreed upon and that must be enforced by the data controller.

[4] The expectation of a data subject in terms of how his or her personal data should be handled.

d. Specifying Authorizations

Data handling policies, preferences, and sticky policies contain, apart from the set of obligations described above, also a set of authorizations. While obligations specify actions that the Data Controller is required to perform on the transmitted information, authorizations specify actions that it is allowed to perform. Similarly to what we did for obligations, we recognize that it is impossible to define an exhaustive list of authorizations that covers all needs that may ever arise in the real world. Rather, we define a generic, user-extensible structure for authorizations so that new, possibly industry-specific authorization vocabularies can be added later on. We do provide however a basic authorization vocabulary for using data for certain purposes and for downstream access control, and we describe how these authorizations can be efficiently matched via a general strategy.

i. Authorization Purposes

The first concrete authorization type that we define is the authorization to use information for a particular set of purposes. Purposes are referred to by standard URIs specified in agreed-upon vocabularies of usage purposes. These vocabularies of URIs may be organized as flat lists or as hierarchical ontologies.

ii. Authorization for downstream usage

The second concrete authorization type that we define is the authorization to forward the information to third parties, so-called downstream data controllers. Optionally, this authorization enables the data subject to specify the access control policy under which the information will be made available, i.e., the minimal access control policy that the (primary) data controller has to enforce when sharing the information with downstream data controllers.

e. Credential requirements

The policy language that we present is geared towards enabling technology-independent user-centric and privacy-friendly access control on the basis of certified credentials. By a credential we mean an authenticated statement about attribute values made by an Issuer, where the statement is independent from a concrete mechanism for ensuring authenticity. The statement made by the issuer is meant to affirm qualification. As credentials are not directly supported in the traditional policy languages, we extended the XACML Rule element such that credentials are the basic unit for reasoning about access control. In our language each rule can contain a Credential Requirements element to specify the credentials that have to be presented in order to satisfy the rule. This element contains a separate Credential element for each credential that needs to be presented. The Credential element can contain restrictions that apply to the credential.

f. Provisional actions

A Provisional Action element is used to specify the provisional actions that a requestor must perform before being granted access to the resource. Currently supported actions include revealing of attributes (to the Data Controller or to a third party), signing a statement, and so-called “spending” of credentials, which allows to put restrictions on the number of times that the same credential is used to obtain access.

g. Data handling policies

Each rule, policy, or policy set can contain a number of data handling policies, each of which is expressed within a Data Handling Policy element. A data handling policy can be referred to from anywhere in the rule by its unique Policy identifier. The main purpose of the data handling policies is for the Data Controller to express what will happen to the information about the Data Subject that is collected during an access request. The provisional action to reveal an attribute value therefore contains an optional reference to the applicable data handling policy. A data handling policy consists of a set of authorizations (described in section 3) that the Data Controller wants to obtain on the collected information, and a set of obligations (described in section 2) that he promises to adhere to. Before the Data Subject reveals her information, these authorizations and obligations are matched against the Data Subject's data handling preferences to see whether a matching sticky policy can be agreed upon.

h. Data handling preferences

The data handling preferences of a rule specify how the information obtained from the resource protected by this rule is to be treated after access is granted. The preferences are expressed by means of a set of authorizations and obligations, just like data handling policies. When access to the resource is requested, the data handling preferences have to be matched against a proposed data handling policy to derive the applicable sticky policy – if a match can be found. An important difference between data handling preferences and data handling policies is the resource that they pertain to: data handling preferences always describe how the resource protected by the rule itself has to be treated, while data handling policies pertain to information that a requester will have to reveal in order to be granted access to the resource. The main use of data handling preferences that we envisage is for a Data Subject to specify how she wants her personal data to be treated by a Data Controller, i.e., which authorizations she grants to the Data Controller with respect to her personal data, and which obligations he will have to adhere to.

i. Sticky policies

The sticky policy associated to a resource, is the agreed-upon sets of granted authorizations and promised obligations with respect to a resource. The sticky policy is usually the result of an automated matching procedure between the Data Subject's data handling preferences and the Data Controller's data handling policy. The main difference between a sticky policy and data handling preferences is that the former contains the authorizations and obligations that the policy-hosting entity itself has to adhere to, while the latter contains authorizations and obligations that an eventual recipient has to adhere to.

Conclusion

In this paper we provide an overview about the different extensions that we propose to add to the standardized policy language XACML in order to enhance its privacy protection capabilities. These improvements correspond to the privacy protection requirements addressed in the PrimeLife project.

Bibliography

[1] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. The Platform for Privacy Preferences 1.0 (P3P1.0)

[2] IBM: Enterprise privacy authorization language (EPAL 1.2)

[3] Moses, T.: OASIS eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard oasis-access control-xacml-2.0-core-spec-os, OASIS (February 2005)

[4] Paschke, A.: ECA-RuleML/ECA-LP: A Homogeneous Event-Condition-Action Logic Programming Language, Int. Conf. of Rule Markup Languages (RuleML'06), Athens, Georgia, USA, 2006

Trabelsi, and Mario Verdicchio. Primelife policy language. 2010. [ACDS08] C.A. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P. Samarati.Â policies and languages. International Journal of Computational Science and Engineering. (IJCSE), 3(2), 2007. PPL (PrimeLife Policy Language) [ABV+09, TNBN10, Par10] is a policy language that can provide access control and usage control. It is specied by PrimeLife (Privacy and Identity Management in Europe for Life) [Prib] which is the follow-up European project of PRIME [PRIa] (Privacy and Identity Management for Europe) aiming to enable privacy protection in the Internet of the citizen in real life. The PrimeLife Policy Language (PPL) is an extension of XACML, thus inheriting its access control capabilities. PPL focuses on the description of the usage of personal data of users by companies. It tackles the following main challenges Policy Requirements PrimeLife collected requirements for data handling, access control, and trust policy languages from the diverse scenarios covered by the project, and analyzed the suitability of existing policy languages to cover the privacy aspects. It quickly became clear that none of the existing policy languages covered all the needs we discovered. It also quickly became clear, however, that satisfying all of the collected requirements was far beyond the available time and budget of PrimeLife.