Can Access Control be Extended to Deal with Data Handling in Privacy Scenarios?

Laurent Bussard
European Microsoft Innovation Center
Aachen, Germany
lbussard@microsoft.com

Moritz Y. Becker
Microsoft Research
Cambridge, UK
moritzb@microsoft.com

In this position paper, we claim that access control policy languages can be extended to address data handling. Indeed, matching users' privacy preferences and services' privacy policies as well as enforcing what services can do with collected data rely on authorization queries and obligations, which exist in some access control languages. We present results from extending SecPAL to address data handling and discuss what could be applied to XACML.

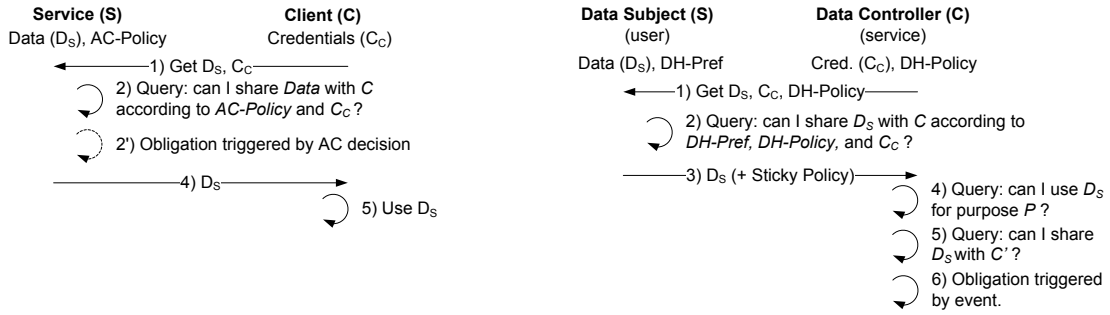
1 Position Statement

We define *data handling* as the possibility to let users (*data subjects*) specify how their personal data should be handled, to let services (*data controllers*) specify how they handle data, and to check whether services' behaviors meet users' requirements. Access control and data handling are both mainly about evaluating authorization queries and enforcing obligations. For instance, queries such as "is this principal authorized to execute this action on this data?" or "can I share this data with this principal?" can appear when enforcing access control as well as data handling. Obligation like "log any read access to this data" can also be part of access control and data handling policies. Even if some queries and some obligations are specific to access control or to data handling, we claim that the same language can be used to specify access control and data handling policies, that the same engine can be used to evaluate queries related to access control and to data handling, and that obligations can be specified and enforced in a unified way.

Unfortunately, state of the art access control policy languages such as XACML [3] and SecPAL [1] cannot be reused straightforwardly to specify data handling because they do not specify obligations and queries in a flexible-enough way. A common language for access control and data handling requires:

1. Specification of access control rules.
2. Evaluation of authorization queries, i.e. deciding whether a principal can execute an action on a given data.
3. Generalized obligations supporting different triggers, i.e. not only triggered by access control decisions.
4. Support for querying obligations, i.e. deciding whether a principal is willing to commit to an obligation.

In this position paper we briefly present "*SecPAL for Privacy*", the result of extending



(a) Access Control

(b) Data Handling

Figure 1: Access Control versus Data Handling.

SecPAL in order to support data handling. A precise description of SecPAL for Privacy and its formal model are available in a research report [2]. We also share general learnings from extending an access control language to address data handling. Finally we discuss the possibility of extending XACML in a similar way to tackle data handling.

2 Similarities and Differences

Fig. 1 compares access control and data handling. Fig. 1(a) presents the access control point of view: client C requests access to data D_S controlled by a service S. S takes its access control policy and C's credentials (C_C) into account to decide whether C is authorized to get D_S. Access control decision may trigger and obligation at S's side. C gets D_S without any explicit constraint on how D_S must be handled.

Fig. 1(b) shows the data handling side: data controller C collects personal data D_S from data subject S. S takes his data handling preferences, C's credentials (C_C), and C's data handling policy into account to decide whether C is authorized to get D_S. D_S is provided to S with explicit constraints (e.g. sticky policy) on how D_S must be handled, which includes rights as well as obligations. The enforcement of the sticky policy leads to evaluating authorization queries and executing obligations at C's side.

Similarities. Let's first look at similarities between access control and data handling: First, queries 2 and 5 of Fig. 1(b) are very similar to query 2 of Fig. 1(a). They are all about deciding whether another party C can get D_S according to S's expectations (AC-Policy or DH-Preferences) and C's properties (credentials and DH-Policy). Second, the "user agent", which evaluates queries 2 and 5 of Fig. 1(b), has a similar role as the "policy decision point" that evaluates query 2 of Fig. 1(a). Finally, in data handling, the policy of C can be considered as a claim describing the behavior of C. From this point of view, data provided by C to S is quite similar in both scenarios.

Differences. There are also important differences between access control and data handling: First, query evaluation in the data handling case is more complex because both sides' rules (i.e. preferences and policies) are taken into account. Second, data handling query 4 of Fig. 1(b) is slightly different than other queries. Third, in access control, obligations are generally triggered by access control de-

cisions. In data handling, obligations may also be triggered by external events (e.g. at a given time). Finally, in data handling, two parties are involved to define obligations, i.e. C promises to S that some obligation will be enforced. In access control, obligations are specified and enforced by one party S.

We rely on similarities and change the access control language to cover differences. The first two differences are solved by extending authorization queries and by specifying matching queries as the evaluation of a set of authorization queries. The remaining differences are solved by specifying generic obligations and related queries.

3 Our Solution: SecPAL for Privacy

The data subject S, i.e. the user, has a collection of pieces of personal data (e.g. D_S). The user specifies *preferences* on handling his personal data. Each data controller C, i.e. each service, specifies a data-handling *policy* on treating the data subject's personal data. During an interaction between S and C, it is checked whether C's policy *satisfies* S's preferences. Assuming that C *complies* with its own policy (i.e. it behaves in accordance to the intended semantics of its policy), then this satisfaction check guarantees that C also complies with S's preferences. Policies and preferences may also specify the conditions when personal data may be *forwarded* to a third party C'. A protocol for data forwarding should ensure that C may only send D_S to C' if C is authorized to forward and if the policy of C' satisfies S's preferences. The framework briefly presented here provides a language for specifying data subject preferences and data controller policies and a method for checking satisfaction between policies and preferences. We refer to [2] for a more formal and comprehensive treatment.

The following gives a more detailed intuition on the intended meaning of preferences and policies. We assume that there is a predefined collection of data handling relevant *behaviors* at the data controller's side, and a corresponding vocabulary for representing these behaviors. These are generally domain-specific, and may include "using an email address for marketing", "forwarding contact details to trusted sellers", "deleting credit card details within one month" or "retaining X-rays for at least 10 years".

The preferences of a data subject S can be divided into two parts. The first part specifies an *upper bound* on a data controller C behaviors with respect to data subject's data (e.g. D_S). It therefore expresses what C is *permitted* to do. The second part specifies a *lower bound* on C's behaviors. It therefore expresses *obligations*, i.e. the behaviors that C must exhibit towards D_S.

A data controller policy can also be divided into two parts. The first part specifies an *upper bound* on its own behaviors. It therefore expresses and advertises the *possible* behaviors of C. The second part specifies a *lower bound* on its behaviors. Therefore, these are *promised* behaviors.

Checking that C's policy satisfies S's preferences consists of two steps. Firstly, every behavior declared as *possible* in the policy must be *permitted* by the preferences. Therefore, it is checked that the upper bound specified in the policy is contained in the upper bound specified in the preferences. Secondly, every behavior declared as *obligatory* in the preferences must be *promised* by the policy. Therefore, it is checked that the lower bound specified in the preferences is contained in the lower bound specified in the policy.

This duality is reflected in the language. The upper bound on behaviors is specified as phrases using the may verb. More specifically,

the upper bound is specified in S's preferences as a collection of may-*assertions*, e.g.

S says ⟨Svc⟩ may use FaxNo for Contact.

In C's policy, the upper bound is specified as a may-*query*, because the corresponding possible behaviors should be a subset of the permitted behaviors. Intuitively, a data controller must ask for permission upfront for anything that it might do with collected personal data in the future. Here is a simple may-*query*:

⟨Usr⟩ says C may use Email for Marketing?

The lower bound on behaviors is specified as phrases using the will verb. More specifically, the lower bound specified by S's preferences is stated in terms of a will-*query*. Intuitively, S asks C to promise the obligatory behaviors. Here is an example of a will-query, in which data subject S requires data controller C to delete his email address within two years:

$\exists t$ (⟨Svc⟩ says ⟨Svc⟩ will delete Email within `t`? $\wedge\ t \leq$ 2 `yr`?)

In C's policy, the lower bound is specified as a collection of will-*assertions*. The following assertion would satisfy the query above:

C says C will delete Email within 1 `yr`.

During an interaction between data subject S and data controller C, placeholders ⟨Usr⟩ and ⟨Svc⟩ are instantiated as S and C respectively. Checking if a service policy satisfies a user preferences is now straightforward. We just need to check if the may-query in the policy and the will-query in the preferences are both satisfied. In general, queries are not satisfied by a single assertion but by a set of assertions. This is because assertions may have conditions that depend on other assertions, and authority over asserted facts may be delegated to other principals. This is why the queries are evaluated against the union of the assertions in the policy *and* the preferences.

4 Discussion

The main technical contributions of SecPAL for Privacy are beyond the scope of this position paper (details in [2]). Comparing the features of SecPAL and XACML as access control languages is also out of scope. In this position paper, we want to share our learnings in extending an access control language to cover data handling and discuss whether and how this could be applied to XACML.

In the introduction, we specify four requirements. Here is how they are covered in SecPAL for Privacy:

1. Specification of access control rules (may-*assertions*).
2. Authorization queries (may-*queries*).
3. Specification of generic obligations (will-*assertions*).
4. Obligation queries (will-*queries*).

XACML partly addresses requirements 1, 2, and 3 but does not cover requirement 4. This is due to the fact that obligations in XACML are specified and enforced within a single trust domain. In the remaining of this section, we discuss what is required in XACML to cover all requirements.

Expressing the upper bound on behaviors (may verb) should be possible with XACML. The data subject's preferences would be a XACML policy and the data controller's preferences would be a set of queries. The user agent would mainly be a Policy Decision Point. Extensions would be required to handle purposes and to specify placeholders that are instantiated before evaluating the query.

Expressing the lower bound on behaviors (will verb) is more difficult. On the data controller's side, it requires a clear specification of XACML obligations and support for obligations that are not triggered by access control

decision. Part of this may be covered by ongoing work on a proposal for obligations[1] in XACML 3.0. Moreover, on the data subject's side, a language to query obligations would be necessary.

Last but not least, upper and lower bounds on behaviors should be expressed in the same language to facilitate reasoning on actions that are authorized and actions that result from obligations.

It is important to note that we only discussed the use of access control engine (e.g. policy decision point) to enforce data handling. We did not tackle cases where the data controller requires data subject's claims, which may be considered as personal data, to enforce access control. For instance, the data subject may have to prove its age to get access to a service offered by the data controller. Having personal data used for access control decision makes the protocol more complex but does not impact our findings.

To conclude, using XACML for expressing users' privacy preferences, services' privacy policies, and sticky policies is not straightforward but seems possible. However, if data handling scenarios are considered as target for XACML, it is important to specify obligations in an appropriate way.

[1]See proposal for obligations (OASIS Wiki) at wiki.oasis-open.org/xacml/ProposalForObligations

Acknowledgments

The authors would like to thanks Alexander Malkis who worked on the specification of SecPAL for Privacy.

Moreover, SecPAL for Privacy is currently evaluated in the PrimeLife project, and part of the evaluation discussed in this paper has received funding from the European Community's Seventh Framework Program (FP7/2007-2013).

References

[1] M. Y. Becker, C. Fournet, and A. D. Gordon. Design and semantics of a decentralized authorization language. In *IEEE Computer Security Foundations Symposium*, pages 3–15, 2007.

[2] M. Y. Becker, A. Malkis, and L. Bussard. A framework for privacy preferences and data-handling policies. Technical Report MSR-TR-2009-128, Microsoft Research, 2009.

[3] OASIS. *eXtensible Access Control Markup Language (XACML) Version 2.0 core specification*, 2005. At `www.oasis-open.org/committees/xacml/`.