Conflict and Combination in Privacy Policy Languages

(Summary)

Adam Barth
Stanford University
abarth@cs.stanford.edu

John C. Mitchell
Stanford University
mitchell@cs.stanford.edu

Justin Rosenstein*
Google, Inc.
justin@roadlesstraveled.org

ABSTRACT

Many modern enterprises require methods for guaranteeing compliance with privacy legislation and announced privacy policies. IBM has proposed a formal language, the Enterprise Privacy Authorization Language (EPAL), for describing privacy policies rigorously. In this paper, we identify four desirable properties of a privacy policy language: guaranteed consistency, guaranteed safety, admitting local reasoning, and closure under combination. While EPAL achieves only one of these four goals, an extended language framework allows us to achieve three out of four, while retaining the basic EPAL framework of restricting access and imposing obligations on users of confidential information.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms

Security, Privacy Policy, Policy Combination

1. INTRODUCTION

In recent years, enterprises have developed privacy policies to assuage customer concerns and to meet legal restrictions on the use of confidential information. These policies are often presented in free text and are difficult to use as a basis for enforcing policy compliance within the enterprise. As a result, an enterprise may not know whether it is in compliance with its own privacy policies, especially in the face of partnering agreements or outsourcing [1].

To enable better enforcement, IBM has introduced the Enterprise Privacy Authorization Language (EPAL) for formally describing policies [2]. An EPAL policy does not itself guarantee conformance, but it can be used in an essential way to achieve conformance. Enterprise members are responsible for respecting restrictions learned from querying an automated authorization service.

In addition to allowing or denying access, EPAL policies can entail "obligations," requiring further actions. For example, an airline company might allow a hotel affiliate to learn a frequent traveller's email address, but then require the affiliate to notify the traveller and allow him or her to opt out of future promotions.

Obligations can be arranged in a hierarchy. For example, the obligation to expunge an itinerary within one week subsumes an obligation to expunge the itinerary within one month. Two obligations might also be incompatible; it is impossible both to expunge an itinerary within one month and to retain the same itinerary for one year. We regard policies that impose incompatible obligations, or both permit and prohibit the same action, as *inconsistent*.

The set of actions governed by a policy also form a natural hierarchy. For example, reading a travel history is tantamount to reading an itinerary because a travel history contains a list of itineraries. Imposing fewer restrictions on reading travel histories than on reading itineraries is *unsafe* because a member of the enterprise can undermine the intent of the policy by reading entire travel histories instead of individual itineraries.

Suppose an airline's policy explicitly states affiliated hotels must check an opt-out list before using email addresses obtained from the airline. A policy auditor may wish to conclude that the entire policy actually enforces this statement, however some languages do not make this a valid conclusion. Policy languages in which such *local reasoning* is justified are more modular and make it easier for policy authors to maintain complex policies that reflect their intent.

Finally, enterprises need to combine privacy policies. For example, an enterprise may wish to enforce privacy legislation and internal policy by combining its own internal policy with a standard industry-wide policy written by a policy provider. Other policy combination issues arise from partnering or outsourcing agreements, as in the JetBlue case study [1]. A policy language is *closed under combination* if it can express the logical combination of any two of its policies. Surprisingly, some policy languages (including EPAL) are not closed under combination.

Ideally, a privacy policy language should force consistency, guarantee safety, permit local reasoning, and allow policy combination. However, these four goals cannot be achieved simultaneously in any language that provides a minimum level of expressiveness.

*This work was done while at Stanford University.

WPES'04, October 28, 2004, Washington, DC, USA.

```
<epal-policy>
  <rule id="a" ruling="allow">
    <user-category refid="Business affiliates"/>
    <data-category refid="Email addresses"/>
  </rule>
  <rule id="b">
    <user-category refid="Affiliated hotels"/>
    <data-category refid="Email addresses"/>
    <obligation    refid="Check opt-out list"/>
  </rule>
</epal-policy>
```

Figure 1: In this abbreviated EPAL policy, statement `b` is never enforced. Interchanging statements `a` and `b` renders the policy unsafe.

2. EPAL: SEQUENTIAL SEMANTICS

EPAL uses a sequential semantics in which the order of statements determines how a query will be answered. Each EPAL statement contains a condition and applies only to queries satisfying this condition. Given a query, the authorization service examines policy statements in order, collecting requirements from applicable statements, stopping after it reaches an applicable statement containing an "allow" or "deny" ruling. Therefore, an EPAL statement is enforced only on those queries satisfying the statement's condition *and* reaching the statement during policy evaluation.

The most compelling reasons for using a sequential semantics is that it forces consistency. Sequential semantics force consistency because policy evaluation is terminated before a conflicting statement could be reached. However, by terminating evaluation mid-policy, EPAL renders sound local reasoning impossible. An unreachable or inapplicable policy statement has no effect on the policy as a whole and therefore could mislead policy authors. For example, the EPAL policy in Figure 1 does not actually require hotels to check the opt-out list. If an affiliated hotel queries the policy regarding use of travellers' email addresses, the authorization service will stop evaluating the policy once it processes statement `a` and, therefore, will not encounter statement `b`, which requires consulting the opt-out list.

Unsafe policies can be written in EPAL. Interchanging the order of statements `a` and `b` results in an unsafe policy: an affiliated hotel could avoid being required to check the opt-out list by querying the authorization service as a generic business affiliate. Unsafe policies are problematic because they allow obligations to be avoided, undermining the policy.

EPAL is not closed under combination. For example, consider the following two compatible EPAL policies. In the first policy, if a member of the marketing department attempts to access a traveller's passport number, then the access is denied and must be logged in marketing's privacy log. In the second policy, if a member of the human resources department attempts to access a traveller's passport number, then the access is denied and must be logged in human resources' privacy log. In the combined policy, if a member of an undetermined department attempts to access a traveller's passport number, then the access is denied and must be logged in both marketing's log and human resources' log. These policies are compatible, but their combination cannot be expressed in EPAL. Proofs, omitted due to length constraint, will appear in another document.

3. DPAL: DECLARATIVE SEMANTICS

We have developed the *Declarative Privacy Authorization Language*, or *DPAL*, a formal policy language that does not terminate evaluation mid-policy. When interpreting a DPAL policy, the authorization service collects requirements from *all* applicable statements, unlike in EPAL. DPAL policies, therefore, enforce each of their statements, enabling both local reasoning and combination. Given a statement from a DPAL policy, an auditor knows the policy enforces the statement without examining the entire policy. Concatenating two policies produces a policy that enforces each statement from each policy. Therefore, in DPAL, concatenation achieves policy combination.

Every EPAL policy can be translated into a DPAL policy, using conditions extended with logical operators. Translating EPAL policies into DPAL enables local reasoning and policy combination, although combined policies might not be expressible in EPAL. Using these extended conditions, DPAL can express the same unsafe policies expressible in EPAL. However, DPAL can be restricted to expressing only safe policies by restricting conditions to be closed upwards.

By not terminating evaluation mid-policy, DPAL allows inconsistent policies to be expressed. However, inconsistencies can be detected algorithmically prior to deployment.

4. CONCLUSION

Ideally, a privacy policy should be consistent and safe, as well as written in a language admitting local reasoning and closed under combination. However, designing languages that force consistency, permit local reasoning, and express all policy combinations is difficult. EPAL forces consistency by terminating evaluation mid-policy, sacrificing safety, local reasoning, and closure under combination.

DPAL does not terminate evaluation mid-policy and thus admits local reasoning, is closed under combination, and can force safety. However, DPAL does not force consistency. Inconsistencies in a DPAL policy can be detected algorithmically, whereas unsound reasoning about an EPAL policy takes place in the mind of the policy's author and, thus, is inaccessible to automated detection.

EPAL policy authoring tools may aid policy authors by translating EPAL policies into DPAL policies in order to identify inapplicable statements. Unfortunately, even those EPAL policies free of inapplicable statements suffer from an inability to admit local reasoning.

Enterprises employing EPAL to express their privacy policies risk deploying policies that do not reflect their authors' intentions. Enterprises employing DPAL can mitigate this risk through the use of local reasoning and combinable modules. Additionally, enterprises using EPAL could be vulnerable to safety-related privacy exploits, whereas enterprises using DPAL need not be. We therefore suggest the unordered language DPAL as an alternative to EPAL.

5. REFERENCES

[1] A. Antón, Q. He, and D. Baumer. The complexity underlying JetBlue's privacy policy violations. *IEEE Security & Privacy*, 2004. To appear.

[2] G. Karjoth and M. Schunter. A privacy policy model for enterprises. In *15th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 2002.