Using Rules to Define the Semantics of Privacy Policies

Grit Denker
David Martin
SRI International, Menlo Park, CA 94025
Grit.Denker@sri.com
David.Martin@sri.com

1 Introduction

Semantic Web technology is a promising approach to a wide variety of Web-based applications in business, finance, government, and education. Typically, these applications involve several parties that do not know each other or have no prior business contacts. It is therefore vital for an entity to know that engaging in a business transaction, a service exchange, or other application will not violate any of its policies. Policies can refer to a broad range of constraints, such as security, authorization, privacy, preferences, contracting, and so on. In this paper, we are interested in investigating privacy policies and determining the suitability of a rule-based approach to privacy policies.

There are several XML-based policy languages, such as Platform for Privacy Preferences Project (P3P) [Cra02], KAoS [UBJ+03], Rei [KFJ05], Enterprise Privacy Authorization Language (EPAL), and XACML. While none of these languages have been particularly studied in the context of rule languages, some of them, such as Rei, express policies in a way that is very close to rules. Our goal is not too compare the different languages and their expressiveness. We are more interested in investigating what language requirements are introduced by privacy policies and how rules can be used to define the meaning of privacy policies. Once we understand the issues involved in expressing the meaning of privacy policies, we can think about translations into existing frameworks or languages to make use of their reasoning and other tool capabilities. Besides the existing language frameworks such as Rei and KAos, that provide various reasoning capabilities, one can also look into the existing rule language engines and systems that might provide a good target for translation of privacy policies.

2 Privacy Policy: Specification and Semantics

In the following, we assume a scenario with two parties jointly engaging in an application. We will refer to the party that initiates the interaction as “requester”, and to the targeted party as “provider”, assuming that the latter provides a certain service that the initiating party would like to execute. Both parties carry their own policies. Exchanging policies is one of the tasks in their interaction. Other tasks are to determine policy compatibility and to monitor and enforce policies during the execution of the service or transaction.

2.1 Privacy Policy Examples

There is a need for privacy policies that define rules and concepts to protect private personal and enterprise information, and to avoid unauthorized collection or distribution

of private data without requesting consent from the owner of the data. The following table gives privacy policy examples.

Example Privacy Policies
[Allow collection of / Will collect] private information.
[Prohibit others to / Will not] collect my private information.
[Do not disclose/ Will not distribute] private information to 3rd parties. [Do not / Will] collect clickstream information
Require keeping private information secret and assuring its integrity
Support encrypted and signed storage of data
Allow keeping of private information for a certain period of time
Require confidential transmission of data

Privacy policies can describe *requirements* ("Require to keep information secret"), *positive and negative authorizations* ("Allow or prohibit to collect information"), or *positive and negative capabilities* ("Support encryption of stored data" or "Cannot sign data"). Positive capabilities can be further specialized to *positive and negative declarations of intent* ("Will collect private information"). Requirements and authorizations are constraints that the advertising party puts on the behavior of the party with which it wants to cooperate. In contrast, capabilities are assertions about the behavior of the advertising party itself, i.e., what it can, cannot, will, or will not do. Thus, for privacy policies we propose to distinguish between these 7 kinds of policies. We define *requirement, posAuth, negAuth, posCap,* and *negCap* as policy classes, and *posIntent* and *negIntent* as subclasses of *posCap.*

Furthermore, we observe that the example policies refer to certain types of actions, namely *data collection*, *storage*, *disclosure*, and *transmission of data*. We propose to define action classes according to these action types, including appropriate subclasses such as *encrypted storage, signed storage, local disclosure, third party disclosure* (with appropriate subclasses for affiliate and non-affiliate third party disclosure), *encrypted transmission, plaintext transmission,* and so on. For example, the *third party affiliation disclosure* subclass is used for rules that allow the disclosure of sensitive information to people, companies, or organizations that have a partnership with the entity.

A party may have a set of policies associated with it. A policy has a certain type and applies to one or more actions and a specific type of privacy information (resource). By default, if no resource is specified, then the rule applies to all types of resources

The policy examples illustrate that there are some distinct characteristics of privacy policies, such as what actions are allowed, required or prohibited on private data. Our approach is to capture these characteristics using an ontology-based approach. We use the Semantic Web language OWL (http://www.w3.org/2001/sw/WebOnt), to define the privacy concepts. A first draft of an OWL privacy ontology that captures some of these concepts and their relationships can be found at http://www.csl.sri.com/users/denker/owl-sec/ontologies/privacy/PrivacySchema.owl.

Assume that a party advertises its policies in the form *policyType(party,actionType,resource)*. E.g., the policy example in the first row of our table can be formalized to *posAuth(?req,dataCollection,?d)* and

posIntent(?prov,dataCollection,?d) for variables *?req*, *?prov*, and *?d* and *Party(?req)*, *Party(?prov)*, and *PrivateData(?d). Party* and *PrivateData* are appropriate classes.

2.2. Describing Policy Semantics

We propose to use a rule-based approach to describe the meaning of policies and their relations. For example, if a party *?p* advertises its intent to encrypt messages *?m*, i.e., *posIntent(?p,encryptedTransmission,?m)*, then one can infer that this party satisfies any policy *pType(?p,Type,?m)* for a policy type *pType* that is a superclass of *posIntent* and for an action type *aType* that is a superclass of *encryptedTransmission.* For example, If *posIntent(?p,encryptedTransmission,?m)* then *posCap(?p,encryptedTransmission,?m)*, meaning that a party intending to encrypt message for transmission has the capability to do so. Thus, we can use rules to describe the relationships between policies. One could use SWRL as the rule language [SWRL03] to define these kinds of relationships. SWRL allows for rules that refer to OWL classes and properties.

Another application of rules is to describe the meaning of policies. For example, as a result of negotiating policies a party might agree to take on an *obligation.* Assume we defined another policy type *obligation* in our ontology. For example, if a requester requires to encrypt all stored information, then a successful collaboration between a requester and a provider would imply that the provider is not only capable of encrypting data, but that he is willing to perform the encryption. A rule capturing this semantics is:

"If *transaction(?req,?prov)* and *requirement(?req,encryptedStorage,?d)* then *obligation(?prov,encryptedStorage,?d)*", where *transaction(?req,?prov)* means that two parties are engaged in a transaction.

Assume that the fact base contains *negCap(?prov,encryptedStorage,?d)*. Negative and positive capabilities are disjoint, that is, if *negCap(?p,?atype,?resource)* and *posCap(?p,?atype,?resource)* then *false.* Since *obligation(…)* implies *posCap(…)*, a reasoner can derive a conflict indicating that provider and requester cannot engage in a transaction without violating one of the requester's policies.

Similarly, the following rule describes that a requester's prohibition to collect implies that the provider will not collect the data:

"If *transaction(?req,?prov)* and *negAuth(?req,dataCollection,?d)* then *negIntent(?prov,dataCollection,?d)*"

These examples show that rules can be useful to define the semantics of privacy policies and establish a basis of reasoning about these policies.

3 Future Work

A similar approach to giving semantics to policies can also be applied to other types of policies, such as authorization and access control policies. In [KPD+04] we used Rei for authorization and confidentiality policy specification. We will investigate how rules can be used to give a formal semantics to those kinds of policies.

More generally, we expect that rules will prove useful to form the basis for a variety of policy related tasks. The technical challenges that must be addressed are:

- What are the requirements on a rule language for specifying privacy policies and their semantics?

- What are the requirements on rule language implementations for analyzing and reasoning about privacy policies?

- What technologies exist or have to be developed to enable enforcement and monitoring of privacy policies?

Depending on the rule language chosen to represent rules defining the semantics of privacy policies, existing tools can support some of these tasks. For example, if we choose a rule language that has a logic programming semantics, policy analysis such as compliance checking will be supported (cf., SweetRules [Sweet]). We are interested in investigating the use of another rule system, namely Maude [Maude]. Maude is a language and engine based on rewriting logic. It allows specifying a wide variety of system structures and behaviors and to execute and validate example scenarios. We envision using Maude as a validation environment for policies in which designers can specify scenarios and check the effect of policies.

4 References

[Cra02] L. F. Cranor. Web Privacy with P3P. Copyright 2002 AT&T. Published by O'Reilly & Associates, Inc., Sebastopol, CA 95472.

[KFJ05] L. Kagal and T. Finin and A. Joshi. Rei: A Policy Specification Language. See http://rei.umbc.edu/.

[UBJ+03] A. Uszok and J. Bradshaw and J. Jeffers and N. Suri and P. Hayes and M. Breedy and L. Bunch and M. Johnson and S. Kulkarni and J. Lott. KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconfliction, and Enforcement. In Proc. of the IEEE Workshop on Policy 2003, IEEE Press.

[SWRL03] I. Horrocks and P.F. Patel-Schneider and H. Boley and S. Tabet and B. Grosof and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. See http://www.daml.org/2003/swrl/rules-all.html.

[Sweet] B. Grosof et al. SWEET - Semantic WEb Enabling Technology. See http://sweetrules.projects.semwebcentral.org/.

[Maude] The Maude System. See http://maude.cs.uiuc.edu/.

[KPD+04] L. Kagal and M. Paolucci and G. Denker and T. Finin and N. Srinivasan and K. Sycara. Authorization and Privacy for Semantic Web Services, IEEE Intelligent Systems, 2004, 19(4):50-56, July/August.