PPL Engine: A Symmetric Architecture for Privacy Policy Handling

Slim Trabelsi[1], Akram Njeh[1], Laurent Bussard[2], Gregory Neven[3]

SAP Labs France, Sophia Antipolis, France[1]
European Microsoft Innovation Center (EMIC), 52072 Aachen, Germany[2]
IBM Zurich Research Center, Zurich, Switzerland[3]

Introduction

Privacy is poorly supported in Web 2.0; in one hand there are, for instance, social network and blogging applications that increase publication of private data that can be linked, shared, aggregated, tagged, and copied. On the other hand mechanisms to control and protect published data are not precise enough. Social networks are is now applying face recognition technology to identify individuals appearing on published photos [1]. At the same time these websites are still publishing a text formatted privacy policy that cannot be read automatically by a browser for example. The privacy protection mechanisms proposed recently in this website are only limited to a very basic access control that concerns the access of the user and not the website owner..Although some existing initiatives like P3P [2], EPAL [3], or XACML [4] already designed some policy based automated solutions to handle separately the access control and the usage control but without a concrete technical deployment. In the context of the European ICT PrimeLife[1] we proposed an extension [5] of the XACML 2.0 called PPL (PrimeLife Policy Language) combining access and usage control policy language. In this paper we describe how the PPL language is deployed, interpreted, and enforced. We developed a solution with following features:

- The language is symmetric and use similar syntax to express privacy preferences of Data Subject (DS), privacy policies of Data Controller (DC), and sticky policies agreed upon by DS and DC. This makes it easier to modify preferences in order to fulfill a policy.
- The architecture is symmetric as well because data subjects and data controllers have similar requirements: deciding whether a given PII (resp. collected data) can be shared with data controller (resp. third party); handling obligations associated with data; storing data and associated preferences (resp. sticky policies). Using the same architecture everywhere to handle scenarios where one party can have multiple roles (e.g. collecting data and next disclosing it to third parties).

Data subject and data controller are obviously different, for instance the data subject can change her privacy preferences regarding her PII while the data controller cannot change the sticky policy associated with collected data. However we show that most of the language and components can be used on each side.

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

PPL Language Structure

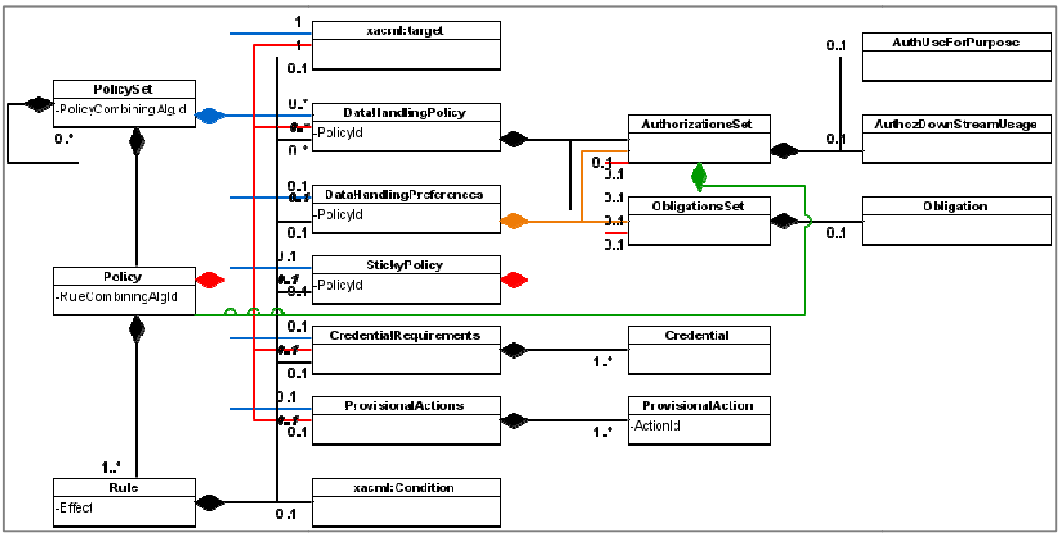

Figure 1 : PPL Policy Structure

The PPL language solution extends XACML with a number of privacy-enhancing, usage control and credential based features. The PPL language is intended to be used by the DC to specify the access restrictions to the resources that he offers. Used by the DS to specify the access restrictions to her personal information and how her data should be treated by the DC afterwards. It is structured as follows:

- **PolicySets**, **Policy**, **Rules**, **Target**, **Condition**: 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. Policies, on their turn, are grouped together in PolicySets; the effect of a PolicySet is determined by the effects of the contained policies and the stated policy combining algorithm. The Target (plain XACML Target), describes the resource, the subject, and the environment variables for which this PolicySet, Policy or Rule are applicable. Finally, the condition specifies further restrictions on the applicability of the rule beyond those specified in the target and the credential requirements.
- **Data Handling Policies**: The main purpose of the data handling policies is for the DC to express what will happen to the information collected from the DS. A data handling policy can be referred to from anywhere in the rule by its unique PolicyId identifier. A data handling policy consists of a set of authorizations, that the DC wants to obtain on the collected information, and a set of obligations, that he promises to adhere to.
- **Data Handling Preferences**: The data handling preferences specify how the data collected by a DC should be treated after the 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.
- **Sticky Policy**: The sticky policy associated to a resource, meaning 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 DS data handling preferences and the DC data handling policy. A part of the sticky policy schema enables the

annotation of the mismatching elements between the DS and DC. This information is only used to display the result of the matching to the user in order to make a decision whether or not the data should be shared. The mismatching information should not appear in the final sticky policy related to a data.

- **Obligation**: Obligations in data handling preferences express actions considered as mandatory by the data subject (e.g. delete collected data within one year). Obligations in data handling policies describe what the service is willing to enforce. Obligations in sticky policies specify what must be enforced. Obligations are specified as triggers and actions, i.e. execute specific actions when given events occur.
- **Authorization**: authorizations specify actions that it is allowed to perform. These actions are split in two: Authorization Purposes that defines 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. The second action is called Authorization for downstream usage and defines the authorization to forward the information to third parties, so-called downstream DC. Optionally, this authorization enables the DS to specify the access control policy under which the information will be made available.
- **Credential Requirements**: 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. This element of the PPL language permits to declare the certified information needed by an entity to get access to a resource. This element is used by the DC in order to express her requirements in terms of certified personal data that should be provided by a DS.
- **Provisional Actions**: A Provisional Action element is used by the DC to specify the provisional actions that a resource requestor must perform before being granted access to the resource. Currently supported actions include revealing of certified and non certified attributes (to the DC or to a third party) under the condition of a specific Data Handling Policy.

PPL Symmetric Architecture

The entire architecture of the PPL engine (see Figure 2) can be represented by 3 layers architecture. The first one presents the user interface layer. The second, the Core layer, represent the main elements of the PPL Engine, which is composed by different subcomponent that we will describe their function below. And the last one represents the persistence layer that is in charge of storing the different data and policies that are used during a transaction between the DC and DS.

Presentation Layer

The presentation layer is responsible of the display to the end user. The presentation layer contains two types of components: the Policy editor that displays and provides a way to manage all the information related to the DS, DC and the third party. This information can be the personal data, the privacy policy/preference, the information involved during a transaction between the different entities, etc. The second component is the matching handler that displays to the user the matching result, by notifying a mismatch in case of, and provide a set of tools that allow him to manage this mismatching. The UI layer is independent from the Core layer. For that, an interface component

might be present between these two layers to provide an abstraction level. The UI is not the same on the DS and DC sides.

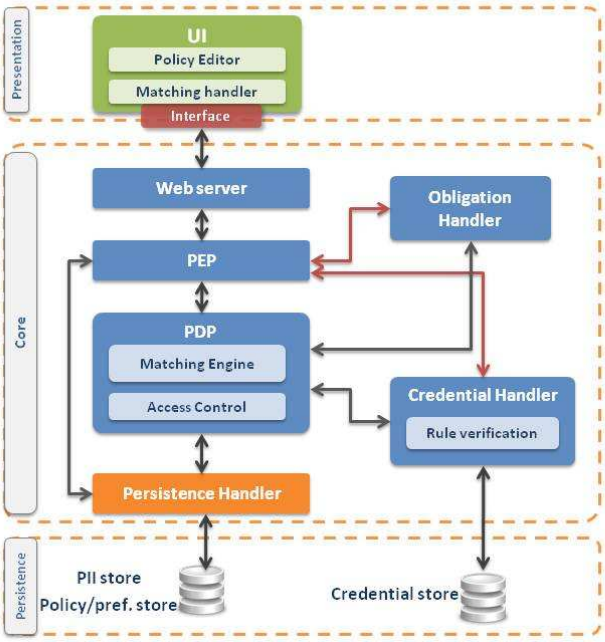

Figure 2 : PPL Engine Architecture

The Core Layer

The Core layer is composed in four main components that are implementing the new concepts introduced within PrimeLife. These components are:

- **Policy enforcement point (PEP)**: This component formats then dispatches the messages to the corresponding component according to the state of the execution process. The decision made by the PDP is enforced in the PEP, meaning that if the PDP decided to provide a data or enforce the access of one resource, this data/resource is collected, formatted and sent to the receiver through the PEP.
- **Policy Decision Point (PDP)**: it is the core of the PPL engine. All the decisions are taken in this component. This latter has two functionalities: the Matching engine that matches between the preferences of the DS and the privacy policy of the DC. The matching is done to verify if the intentions of the data controller in terms of private data usage are compliant with the data subject preferences. The Access control engine is in charge of the enforcement of the access control rules related to the local resources. It analyses the resource query, check the access control policy of the requested resource and decides whether or not requester satisfies the rule.
- **Credential Handler**: one of the new features introduced in PPL is the support of the credential based access control. This feature is implemented by the credential handler that manages the collection of credential held by an entity, selects the appropriate credentials in order to generate a cryptographic proof and verifies the cryptographic proofs of the claims received from external entities. The credential handler component contains the subcomponent Rule Verification; the PPL policy contains a description of the credential requirements (for access control), the Rule Verification component evaluates whether the

claim provided by a user that wants to access a resource satisfies the credential based access control rule.

- **Obligation handler**: is responsible for enforcing the obligations that have to be satisfied by the DC. This engine executes two main tasks: setup the triggers related to the actions required by the privacy preferences of the Data Subject, and executes the actions specified by the data subject whenever it is required.

The other components of the Core layer play a secondary role in the concept introduced by the PPL engine like the Web server that is an embedded server that represents the entry point of the core of the PPL Engine. It can be seen as an interface to the PEP. The persistence handler which is the interface between the Core and persistence layer. It makes transparent to the Core layer location and storage model of the data it manipulates. In general, this layer is supported by a Persistence Framework. The defined objects in this layer are generally DAO (Data access Object). The persistence handler provide management functions to handle the DAO know as CRUD (create, retrieve, update, delete) methods.

Persistence Layer

The persistence layer is represented by the Data/Policy store contains all the information on private data and their related policies and by the credential store which contains all the credentials and the certified information held by an entity. The access to this store is exclusively allowed to the Credentail Handler component.

Conclusion

In this paper we show how the PPL Privacy Policy engine is implemented. We rely on a symmetric architecture that fits with the current data utilization model where personal data is shared and stored by multiple parts without a real control from the owner. According to the location of the data, the engine can react as a data owner or data collector and execute the appropriate tasks.

Bibliography

[1] Sarah Perez, “Facial Recognition Comes to Facebook”, http://www.readwriteweb.com/archives/facial_recognition_comes_to_facebook.php

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

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

[4] 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)

[5] Claudio A. Ardagna, Eros Pedrini, Sabrina De Capitani di Vimercati, Pierangela Samarati, Laurent Bussard, Gregory Neven, Franz-Stefan Preiss, Stefano Paraboschi, Mario Verdicchio Dave Raggett, Slim Trabelsi, “PrimeLife Policy Language”, W3C Workshop on Access Control Application Scenarios, November 2009, Luxembourg.