Data Usage Control in the Future Internet Cloud

Michele Bezzi and Slim Trabelsi

SAP Labs,
06253, Mougins, France

Abstract. The increasing collection of private information from individuals is becoming a very sensitive issue for citizens, organizations, and regulators. Laws and regulations are evolving and new ones are continuously cropping up in order to try to control the terms of usage of these collected data, but generally not providing a real efficient solution. Technical solutions are missing to help and support the legislator, the data owners and the data collectors to verify the compliance of the data usage conditions with the regulations. Recent studies address these issues by proposing a policy-based framework to express data handling conditions and enforce the restrictions and obligations related to the data usage. In this paper, we first review recent research findings in this area, outlining the current challenges. In the second part of the paper, we propose a new perspective on how the users can control and visualize the use of their data stored in a remote server or in the cloud. We introduce a trusted event handler and a trusted obligation engine, which monitors and informs the user on the compliance with a previously agreed privacy policy.

Keywords: Privacy, Usage control, Privacy Policy

1 Introduction

The vision of the Future Internet heralds a new environment where users, services and devices transparently and seamlessly exchange and combine information, giving rise to new capabilities. In order for it to materialize, this vision needs a mix of adaptation of existing technologies and business models, such as flexible infrastructures and service compositions, distributed ownerships, and large-scale collaborations. The *cloud* is one of the first instantiations of these paradigms. In the cloud users and businesses can buy computing resources (e.g., servers, services, applications) provided by the cloud, that are rapidly provisioned with a minimal management effort and pay-per-use. In the cloud, data may flow around the world, ignoring borders, across multiple services, all in total transparency for the user.

However, this ideal cloud world raises concerns about privacy for individuals, organizations, and society in general. In fact, when data cross borders, they have to comply with privacy laws in every jurisdiction, and every jurisdiction has its own data protection laws. In addition, the risk, for personal data to travel across boundaries and business domains, is that the usage conditions agreed

J. Domingue et al. (Eds.): Future Internet Assembly, LNCS 6656, pp. 223–231, 2011.

upon collection are lost, and, as a consequence, users cannot control their personal information any more, as well as, honest businesses may lose confidence in handling data, when usage conditions are uncertain.

To face these challenges, the concept of *sticky policy* has been introduced [5]. Personal information is associated with a machine-readable policy (*sticky policy*), which stipulates the ways and means to treat that information (for example, expressing that the data should be used for specific purposes only, or the retention period should not exceed 6 months, or the obligation to send a notification to the user when data are transfered to a third party). The sticky policy is propagated with the information throughout its lifetime, and data processors along the supply chain of the cloud have to handle the data in accordance with their attached policies.

The concept of sticky privacy policy represents a powerful instrument to address many privacy requirements. However, its application requires that several problems be solved:

- Expressing privacy policy in a machine-readable language. Although various policy languages have been introduced so far [7,1,2], there is no single language able to completely address the most important privacy scenarios, such as setting and comparing user preferences with server privacy policies, expressing conditions on complex secondary usage cases, specifying obligations and integrating access control policies.
- Providing the data owner with a user-friendly way to express their preferences, as well as to verify the privacy policy the data are collected with.
- Develop mechanisms to enforce these sticky policies in ways that can be verified and audited.

In this paper, we present recent results obtained by the European ICT project PrimeLife which (partly) addresses these problems, introducing a novel policy language to express complex privacy conditions, and the corresponding policy engine able to process these policies. In particular, in Sect. 2 we introduce the PrimeLife Policy Language (PPL), which combines access and data handling policies; we then describe the corresponding policy engine, enabling the deployment, interpretation and enforcement of PPL policies. Although the proposed solution can address the main requirements to manage privacy policies in the cloud, there are still important open problems to address (see Section 3). In particular, the current framework lacks mechanisms to provide the data owner with the guarantee that policy and obligations are actually enforced. In Sect. 4, we present our initial thoughts on how to implement a trusted system for policy enforcement. Conclusions are drawn in the last section.

2 Primelife Privacy Framework

In many web applications, users are asked to provide various kinds of personal information, starting from basic contact information (addresses, telephone, email) to more complex data such as preferences, friends' list, photos. Service providers

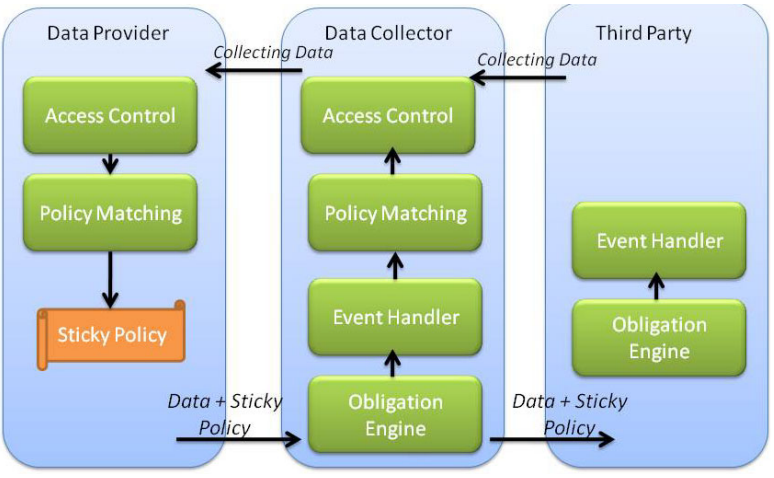

Fig. 1. PPL high level architecture.

describe how the users' data are handled using privacy policy, which is, more or less explicitly, presented to users during the data collection phase. Privacy policies are typically composed of a long text written in legal terms that are rarely fully understood, or even read, by the users. As a result, most of the users creating accounts on web 2.0 applications are not aware of the conditions under which their data are handled.

Therefore, there is need to support the user in this process, providing an as-automatic-as-possible means to handle privacy policies. In this context, the European FP7 project PrimeLife[1] developed a novel privacy policy framework able to express and automatically process privacy policies in web interactions. This approach enables applications, like web browsers, to automate the interpretation of the content of a privacy policy and to compare the service privacy policy with user privacy preferences.

The Primelife project introduced the PrimeLife Policy Language (PPL, herein) [10,4], which allows to describe in an XML machine-readable format the conditions of access and usage of the data. A PPL policy can be used by a service provider to describe his privacy policies (how the data collected will be treated and with whom they will be shared), or by a user to specify his preferences about the use of his data (who can use it and how it should be treated). Before disclosing his personal information, the user can automatically match his preferences with the privacy policy of the website and the result of the matching generates an agreed policy, which is bound to the data (sticky policy) and travels with them. In fact, this sticky policy will be sent to the server and follow the data in all their lifecycle to specify the usage conditions.

The PPL sticky policy defines the following conditions:

[1] www.primelife.eu

- Access control: PPL inherits from the XACML [8] language the access control capabilities that express how access to which resource under which condition can be achieved.
- Data Handling: the data handling part of the language defines two conditions:
  - Purpose: expressing the purpose of usage of the data. Purpose can be for example marketing, research, payment, delivery, etc.
  - Downstream usage: supporting a multi-level nested policy describing the data handling conditions that are applicable for any third party collecting the data from the server. This nested policy is applicable when a server storing personal data decides to share the data with a third party
- Obligations: Obligations in sticky policies specify the actions that should be carried out after collecting or storing a data. For example, notification to the user whenever his data are shared with a third party, or deleting the credit card number after the payment transaction is finished, etc..

Introducing PPL policies requires the design of a new framework for the processing of such privacy rules. In particular, it is important to stress that during the lifecyle of personal data, the same actor may play the role of both data collector and data provider. For this reason, PrimeLife proposed the PPL engine based on a symmetric architecture, where any data collector can become a data provider if a third party requests some data (see Figure 1). According to the role played by an entity (data provider or data collector) the engine behaves differently by invoking the appropriate modules.

In more detail, on the data provider side (user) the modules invoked are:

- The access control engine: it checks if there is any access restriction for the data before sending it to any server. For example, we can define black or white lists for websites with whom we do not want to exchange our personal information.
- Policy matching engine: after verifying that a data collector is in the white list, a data provider recovers the server's privacy policy in order to compare it to its preferences and verify whether they are compatible in terms of data handling and obligation conditions. The result of this matching may be displayed through a graphical interface, where a user can clearly understand how the information is handled if he accepts to continue the transaction with the data collector. The result of the matching conditions, as agreed by the user, is transformed into a sticky policy.

On the data collector side, after recovering the personal information with its sticky policy the invoked modules are:

- Event handler: it monitors all the events related to the usage of the collected data. These event notifications are handled by the obligation engine in order to check if there is any trigger that is related to an event. For example, if a sticky policy provides for the logging of any information related to the usage of a data, the event handler will notify the obligation engine whenever an

access (read, write, modification, deletion etc.) to data is detected in order to keep track of this access.
- Obligation engine: it triggers all the obligations required by the sticky policy.

If a third party requests some data from the server, the latter becomes a data provider and acts as a user-side engine invoking access control and matching modules, and the third party plays the role of data collector invoking the obligation engine and the event handler

3 Open Challenges

Although the PPL framework represents an important advancement in fulfilling many privacy requirements of the cloud scenario, there are still some issues, which are not addressed by the PPL framework.

Firstly, in the current PPL framework, the data owner has no guarantee of actual enforcement of the data handling policies and obligations. Indeed, the data collector may implement the PPL framework, thus having the technical capacity of processing the data according to the attached policies, but it could always tamper with this system, which controls, or simply access directly the data without using the PPL engine. In practice, the data owner should trust the data collector to behave honestly.

A second problem relates to the scalability of the sticky policy approach. Clearly, the policy processing adds a relevant computational overhead. Its applicability to realistic scenarios, where large amounts of data have to be transmitted and processed, has to be investigated.

A last issue relates to the privacy business model. The main question is: What should motivate the data collectors/processors to implement such technology? Actually, in many cases, their business model relies on the as-less-restricted-as-possible use of private data. On the user side, a related question is, are the data owners ready to pay for privacy [9]? Both questions are difficult to address, especially when dealing with such a loosely defined concept as privacy. Although studies exist (see [11,3], and references therein), mainly in the context of the web 2.0, we should notice that the advent of cloud changes the business relevance of privacy. In fact, in a typical web 2.0 application the user is disclosing his own data, balancing the value of his personal data with the services obtained. As a matter of fact, users have difficulties to monetize the value of their personal information, and they tend to disclose their data quite easily. In the cloud world, organizations store the data they have collected (under specific restrictions) with the cloud provider. These data have a clear business value, and typically companies can evaluate the amount of money they are risking if such data are lost or made public. For these reasons, it is likely that they are ready to pay for a stronger privacy protection.

All these issues need further research work to be addressed. In the next section, we present our initial thoughts on how we may extend the Primelife framework to address the first problem we mentioned above, i.e., how to provide a secure enforcement for privacy policy.

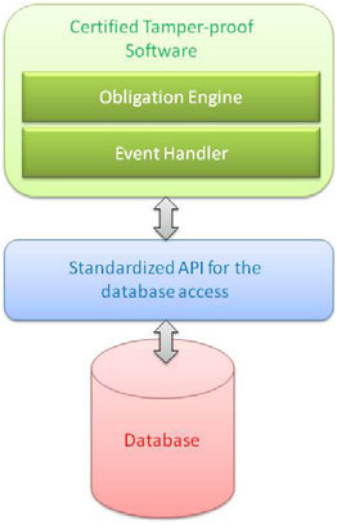

Fig. 2. The key elements of the extension of the PPL framework to guarantee the enforcement of privacy policy.

4 Towards Privacy Policy Enforcement in the Cloud

In the current PPL framework, there is no guarantee of enforcement of the data handling policies and obligations. In other words, we suppose that the server enforces correctly the sticky policies, but, actually, nothing prevents him from creating a back door in his database in order to get unauthorized access to the collected information.

For this reason, we propose in the rest of the paper a secure architecture for the enforcement of the sticky policies and facilitating the task of external auditors to verify the compliance with the privacy requirements, as well as giving the user control on the released data. The main idea is to introduce tamper-proof [6] obligation engine and event handler, certified by a trusted third party, which mediate the communication and the handling of private data in the cloud platform. The schedule of the events, as well as the logs of these components can also be (partly) accessed by the users to monitor the handling of their personal information. Lastly, the trusted-third party can ensure the auditing of the whole system.

Let us sketch how our proposal can work in a simple cloud scenario. Let us consider a cloud platform provider, which hosts one or more services/applications provided by external parties that deal with personal data (e.g., a human resource management application, a remote storage service). Say, these services handle personal data using a PPL framework (as described in Sect. 2). In order to guarantee enforcement of the privacy policies and corresponding obligations by the service, we replace the service provider obligation engine and event handler

with a tamper-proof event handler and a tamper-proof obligation engine certified by a trusted third party (e.g., governmental office), see Fig. 2. For instance, the cloud provider may provide these certified components as premium service.

In fact, trust is an essential part of the cloud paradigm. If the data owner has the guarantee from a trusted authority (governmental office, EU commission, etc.) that the application hosted in the cloud is compliant with his privacy requirements, he will tend to transfer his data to the certified host. In order to certify the compliance of an application, the trusted authority has, first, to certify the secure privacy components in charge of enforcing sticky policies, then to perform audits to check if the stored data are handled correctly.

The difficulty comes for the access to the database by the service provider. One solution would be to use a specific tamper-proof database, but this can be technically complex, and impact the business efficiency of the service provider. A possible solution is to specify an API to access the database that is compatible with the event handler. This API should be defined as a standard interface to communicate with the event handler and access to the database. The service has to exclusively use an interface compatible with the standardized API, and this should be subject to audit by an external trust authority (which could be the same or not certifying the tamper proof components).

Data URI	Data Type	Sticky Policy	Pending Obligations	Related Events			Admin actions
				Type	Date	Time	
#12345	e-mail	SP#12345	Delete in 5 minutes	Read	02/10	11:34	Del/Move

Fig. 3. A sketch of data track administration console

The particularity of this API is that all the methods to access the data can be detected by the event handler. For example, if the service adds a new element (data and sticky policy) this action should be detected, managed and logged by the event handler. If there is any method (like table dump) to access the database that cannot be recognized by the event handler, the service will not be certified by the trusted authority.

Using a tamper proof event handler and obligation engine also gives the possibility of providing a monitoring console. The monitoring can be accessible by any data owner, who, once authenticated, can list all the data (or set of data) with their related events and pending or enforced obligations. The data owner can at any time control how his data are handled, under which conditions the information is accessed, and compare them with the corresponding stored sticky policy. Fig. 3 shows a very simple example of how the remote administrative console could be structured, this monitoring console could of course be more complex. The remote monitoring console adds more transparency and more control to the data hosted within the cloud. It also allows the user to detect any improper usage of his data, and, in this case, notify the host or the trusted authority.

The advantages of the proposed solution are twofold. First, from the data owner perspective, there is a guarantee that actual enforcement has taken place, and that he can monitor the status of his data and corresponding policies. Second, from the auditors' point of view, it limits the perimeter of their analysis, since the confidence zone provided by the tamper proof elements and the standardized API facilitate the distinction between authorized and non authorized actions.

5 Conclusions

Cloud computing and the SOA paradigm are fundamental building blocks for the Future Internet, enabling the seamless combination of services across platforms, geographies, businesses and transparently from the user point of view. However, these new capabilities may entail privacy risks. From the user perspective, the risk is that of losing control of his personal information once they are released in the cloud. In particular, when personal data are consumed by multiple services, possibly owned by different entities in different locations, the conditions of the data usage, agreed upon collection, may be lost in the lifecycle of the personal data. From the data consumer point of view, businesses and organizations seek to ensure compliance with the plethora of data protection regulations, and minimize the risk of violating the agreed privacy policy.

The concept of sticky policy may be used to address some of the privacy requirements of the cloud scenario. In this paper we reviewed the recently introduced PPL framework, which provides a flexible language to express privacy policy as well as the necessary mechanisms to process and compare sticky policies. The current PPL framework presents some limitations; it notably requires a high level of trust in the data collector/processor. We presented some initial thoughts about how this problem can be mitigated through the usage of a tamper proof implementation of the architecture. This solution may increase the trust on the cloud, but it still needs further studies to verify its applicability in real life business scenarios.

Acknowledgements. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216483.

References

1. Ardagna, C.A., Cremonini, M., De Capitani di Vimercati, S., Samarati, P.: A privacy-aware access control system. J. Comput. Secur. 16, 369–397 (2008)
2. Ashley, P., Hada, S., Karjoth, G., Powers, C., Schunter, M.: Enterprise privacy authorization language (EPAL 1.1). IBM Research Report (2003)

3. Bonneau, J., Preibusch, S.: The privacy jungle:on the market for data protection in social networks. In: Moore, T., Pym, D., Ioannidis, C. (eds.) Economics of Information Security and Privacy, pp. 121–167. Springer, New York (2010)
4. Bussard, L., Neven, G., Preiss, F.S.: Downstream usage control. In: IEEE International Workshop on Policies for Distributed Systems and Networks, pp. 22–29 (2010)
5. Karjoth, G., Schunter, M., Waidner, M.: Platform for enterprise privacy practices: Privacy-enabled management of customer data. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 69–84. Springer, Heidelberg (2003)
6. Naedele, M., Koch, T.E.: Trust and tamper-proof software delivery. In: Proceedings of the 2006 international workshop on Software engineering for secure systems. SESS '06, New York, NY, USA, pp. 51–58. ACM Press, New York (2006), doi:10.1145/1137627.1137636
7. Reagle, J., Cranor, L.F.: The platform for privacy preferences. Commun. ACM 42, 48–55 (1999), doi:10.1145/293411.293455
8. Rissanen, E.: extensible access control markup language (xacml) version 3.0, extensible access control markup language (xacml) version 3.0, oasis (August 2008)
9. Shostack, A., Syverson, P.: What price privacy? In: Camp, L., Lewis, S. (eds.) Economics of Information Security, Advances in Information Security, vol. 12, pp. 129–142. Springer, New York (2004)
10. Trabelsi, S., Njeh, A., Bussard, L., Neven, G.: The ppl engine: A symmetric architecture for privacy policy handling. W3C Workshop on Privacy and data usage control p. 5 (October 2010), http://www.w3.org/2010/policy-ws/
11. Tsai, J.Y., Egelman, S., Cranor, L., Acquisti, A.: The Effect of Online Privacy Information on Purchasing Behavior: An Experimental Study. In: ICIS 2007 Proceedings, p. 20 (2007)