Privacy-Aware Service Integration

Pierre Parrend, Stéphane Frénot, Sebastian Hoehn

Privacy-Aware Service Integration

Pierre Parrend, Stephane Frenot
INRIA ARES/CITI, INSA-Lyon
21, Av. Jean Capelle
F-69621 Villeurbanne, France
pierre.parrend@insa-lyon.fr
stephane.frenot@insa-lyon.fr

Sebastian Höhn
Albert-Ludwig University
Institute of Computer Science and Social Studies
Dept. of Telematics
Friedrichstr. 50
79098 Freiburg, Germany
sebastian.hoehn@iig.uni-freiburg.de

Abstract

Privacy mechanisms exist for monolithic systems. However, pervasive environments that gather user data to support advanced services provide little control over the data an individual releases. This is a strong inhibitor for the development of pervasive systems, since most users do not accept that their personal information is sent out to the wild, and potentially passed over to third party systems.

We therefore propose a framework to support user control over the data made available to service providers in the context of an OSGi based Extensible Service Systems. A formal privacy model is defined and service and policy descriptions are deduced. Technical system requirements to support these policies are identified. Since guaranteeing privacy inside the system is of little help if any malicious entity can break into it, a security architecture for OSGi based Extensible Service Systems is also defined.

1 Pervasive Service Systems

In order to evolve from rigid systems or research tools towards dynamic service environments suitable for commercial systems, pervasive systems need to provide security and privacy during service discovery and execution. Users do not want their private data to be passed on to third parties, or extensive profiling to be performed on the basis of the data they send to perform single requests. Such a control is extremely difficult to achieve in the case of open pervasive systems, *i.e.* systems where unknown providers can offer their services. Fortunately, this generic case does not match the current business models of pervasive systems. The latter can be defined as follows: one Service Provider is responsible for making services available to users and for ensuring that they are allowed to access them. The master Service Provider typically calls on subcontractors to advertise a large set of services. He is responsible for the publication of services and for the quality of service (and hence also privacy and security requirements).

Among possible use cases are automotive telematic systems [5, 12], which allow drivers to find their way using real time traffic information, to be informed of nearby restaurants or to use on-board information systems. A second example are Home Gateway systems [9], which let users manage their home appliances and install new software extensions. A third is the retail store, which uses RFID chips to track supplies bought by the customers and provides additional commercial information or for instance vocal information on available products for the visually impaired.

The services in such pervasive systems are made available under two complementary forms: either remotely accessible services (e.g. Web Services), or downloadable (OSGi) code bundles, which provide services that are locally accessible. These we call *Pervasive Service Systems*. Privacy can be achieved in this context by controlling the data that is collected by all (remote and local) services and the propagation of this data. Therefore, the privacy properties of each service must be published as service meta-data. Trustworthiness of meta-data can be achieved in business scenarios because one single provider is responsible for publishing the services and their meta-data. The providers have therefore a strong incentive to publish correct declarations, in order to maintain their business relationships. Furthermore, meta-data can be considered a legal contract and law suits are initiated if these contracts are not respected [14].

The goal of this paper is to provide a framework for monitoring the privacy properties of Pervasive Service Systems, along with the technical requirements it implies. The framework is a compound of two elements: an architecture for secure interactions between the users and the pervasive system and a meta-data language that expresses privacy properties of services and user-defined policies.

This paper is set out as follows. Section 2 presents the security architecture for Pervasive Service Systems and section 3 the privacy model for such systems. Section 4 discusses the technical requirements of the end-user execution platform for privacy policy enforcement.

2 Secure Architecture for Pervasive Services Provisioning

The security architecture is a strong requirement for privacy-aware systems: if an attacker can gain access to personal information all other countermeasures become useless.

First, the service publication and provisioning processes are defined. Secondly, a security analysis of this architecture is performed.

2.1 Architectural Overview

Highly dynamic systems need to support runtime extension: in the case of automotive services, the change of localization often implies the modification of available services; in the case of Home Services, new devices are bought by the users and their functionalities need to be supported; in the case of the Future Supermarket, the user can go to one or another store, or specific shelves may be bound to specific services within the same store. Two complementary mechanisms can be used to support the runtime extensibility of such systems: (1) the discovery of remote services (as in SOA systems), and (2) the runtime extension feature provided by the OSGi platform, which makes it possible to install new applicative *bundles* without perturbating the running services (in particular without the need to reboot).

Two steps are necessary to enable the end-user take advantage of the services: the publication of those services (in the form of bundles or of remote services), and their provisioning by the user. The provisioning means the process of discovery, selection, validation and integration of the services in the user system. The end-user can either act directly to obtain new services or the system performs this operation automatically to update the services, adapt them to an evolving context or ensure the continuity in front of changing services.

Two protocols are defined: one for the discovery and provisioning of remote services and the other for bundles. The protocols are parallel: the client system gets the list of available services and automatically selects the ones that are compliant with local policies. The users can select the services/bundles they actually want to use. In the case of bundles, these latter are downloaded and installed. The applications can then be used. Figure 1 shows this protocol in the case of OSGi bundles.

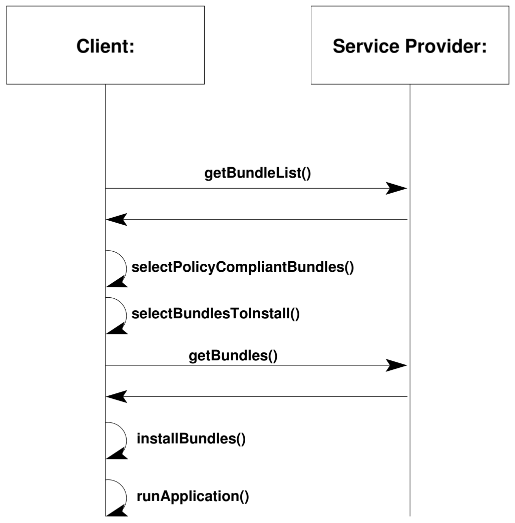

Figure 1. Bundle Discovery and Provisioning

The monitoring of privacy in highly dynamic systems is based on two fundamental requirements. First, the architecture needs to be protected against external aggressions and malicious actors. Then, a framework for ensuring policy-compliant service behavior inside the system is required.

2.2 Security Analysis

The protection of the Pervasive Service Architecture is achieved through two complementary elements: a service supplier side security control and a client side security control. The security modules on the client execution platform are also presented. The reference security properties for a distributed system are confidentiality, integrity and availability [7]. These properties must be enforced on the supplier side as well as on the client side.

The service providers must ensure that the services they propose use secure communication channels for service access and for bundle deployment. This enforces the property of confidentiality and integrity of the communication against eavesdropping and man-in-the-middle attacks. It must also guarantee that the users that access the services are allowed to do so. Next, to limit the potential reliability weaknesses that could result in security breaches on the client side, the provider must guarantee that the remote services and bundles do not introduce reliability weaknesses. This is typically achieved through extensive testing or use of fuzzing techniques, *i.e.* giving random data as service input to stress them. Lastly, the service provider is responsible for providing security and privacy meta-data for the services. In the considered use cases, a single service provider manages the system. It is considered as trusted with regard to meta-data publication, since the commercial service provider will not provoke inconsistency in the behavior of services.

The client execution environment must supervise these properties on its own side. The communication channel must be protected to ensure confidentiality and integrity. The remote servers must be authenticated to prevent the use of malicious services or installation of malicious bundles. Moreover, the client platform must check the compliance of advertised services with local security policies. Lastly, the client platform should have satisfactory reliability properties and be robust against ill-formed bundles and services.

3 Privacy Model

The Privacy Model for Pervasive Service Systems supports the expression of the data available and of their manipulation. It is based on a formal representation, and implemented as two complementary data sets: the privacy related behavior of the services, and the user policies that describe the acceptable behavior of the services used.

3.1 Formal Foundation

Several formal models for describing privacy requirements and obligations have been developed. The most adequate ones [1, 3] are based on temporal logic and are very well suited to describe privacy requirements in general. An integration of two things is currently missing: (1) attributes and their association to individuals and (2) the context of the administrative domains the different service are operating in.

Attributes are at the center of privacy and hence its formal models because each data profile consists of a collection of attributes from a given individual. What becomes more important in service-oriented systems is distribution of the services and their providers. Current models focus on an e-business scenario where one provider interacts with the users and collects, stores and processes data. This bilateral relationship is no longer common to the upcoming architectures: services are provided by a plethora of providers and a plethora of services from one provider is consumed. So the underlying privacy model must be adapted to this new scenario. The different providers and their administrative domain must become part of the model. Privacy threats in general can only arise from the providers' view on the data, if we assume that it is too costly to merge databases that arise from different administrative domains or if we deploy protocols to ensure unlinkable transactions [15].

The different entities of this privacy model is a finite set Id of users identified by a unique name. A set of local actions Act_i details the way these users interact with the system. The information that is collected about a specific user is a set of attributes A. This set is allocated to the individuals in a given context. That means there exists a relation $Id \times A$ which associates these attributes to the individuals.

If we consider the consumers' point of view in this approach, we need a notion of administrative domains. Other models lack this point of view because they either take the service providers' point of view or a global one. Both of them are not very well suited to describe privacy requirements of individuals in a rather complex scenario. An administrative domain encompasses a collection of attributes that the individuals share with this domains' systems. If we consider two services that are provided by the same service provider, we realize that the attributes shared with each of the services can easily be linked and added to the profile by the provider. Although the architecture encompasses a single "master provider" that guarantees the quality of the meta-data presented to the user, this provider will certainly have subcontractors fulfill certain tasks. These subcontractors are to be modeled very closely in order to make statements on the users' privacy policies. There exists a finite set of administrative domains D. Each of these domains $d \in D$ has a dataset κ_d, which consists of the attributes the systems of this domain have collected over time. κ_d is a subset of $a \times Id$, the set of attributes of given users. There may also exist a membership relation between the individuals and the administrative domains: the set of members of an administrative domain M_d is a subset of $D \times Id$.

Each action Act_i that is performed takes place in an administrative domain d. We denote this as $@_d[Act_i]$. This is important because individuals might change their membership within an administrative domain and hence can spread their knowledge.

The dataset of a given domain is not static. As the members of the different domains start interacting with each other, i.e. they share messages on different attributes of individuals, they spread these attributes throughout the different datasets. Furthermore it is possible for the providers of the datasets to derive new information from a given subset of attributes. For example, if we know the birthday of an individual, we might easily calculate its age. In real scenarios this is a lot more complex: we assume that attributes are rather complex and possibly contain unique identifiers. These complex attributes can be linked together and the evolution of these attributes over time reveals new information, i.e. additional attributes of the individuals. For deriving new attributes from the current dataset we define a set of production rules for each of the datasets. Where each of these rules intuitively consists of an "if part" and a "then part". The "if part" is P_{if} is a subset of the attributes P_i of an individual i: $P_{\text{if}} \subseteq P_i$. The "then part" is one of the individual's attributes a_i. The set of the production rules is hence defined as $R_p \subseteq P_{\text{if}} \times a_i$.

3.2 Implementation of Meta-data and Policies

The formalization presented above brings us into a position where we can express the parts of a service-oriented system: (1) services are the actions (2) an individual user can perform. (3) data attributes and (4) administrative domains are explicitly modeled. Together with the production rules and formulae given in an adequate logical representation, we can express privacy requirements. Production rules are necessary because we must assume that the attacker will use data mining techniques to aggregate the information that is needed for a given exploit.

To efficiently establish the model, meta-data must be reduced to a smaller number of possible elements. It is not reasonable, though formally correct, to define each of the services as one action. This definition would not allow for a semantic interpretation of the model unless a detailed service description is provided with each of these services. Since the evaluation of the impact on the users' privacy cannot be inferred automatically (unless these descriptions are modeled once again and we get into a recursive situation here), we do not get one step ahead from the current situation (where we have written privacy policies and someone must read them and judge the impact on the users' privacy). To this end, it is necessary to define a subset of actions that are suited to harm the users' privacy. This is reading, writing, modification and sending of personal data. Each of the services contains in its meta-data the actions it performs with the personal data.

The same considerations hold for the data attributes. Data attributes are a lot more complicated because they are domain specific. We will certainly not be able to define a generic set of attributes that can be collected on people. To this end, it is necessary to define an application specific set of attributes. This is not a new issue as this definition has successfully been implemented in the area of identity management for several years [15]. The predefined set of attributes then has a clear semantics to the end users (e.g their birthdays) and they can easily estimate the impact their disclosure might have.

Administrative domains are once again rather generic. There exists a set of domains (for example, named according to their providers). Each of these domains has a distinct level of trust. This trust is specific for each of the users and cannot (and need not) be part of the model. Trust depends on many factors, for example, if a contract between the user and the provider exists or a subcontract from the "main provider" only.

We can then provide the users with a realistic estimation of the data profiles saved with each of the different providers. This information must be visualized in an adequate manner because it can be rather complex. A summary can easily be prepared if we once again consider the partial identities we gathered from the identity management mechanisms we have at hand. As long as the data distribution corresponds to the partial identities provided by the user, there is no need to react. As soon as one of the providers is able to bring together different attributes from different interactions and establish a profile that is larger than the partial identities used by the client, there is possibly a need to react (or at least a need to review the situation).

For automating the analysis of privacy requirements, the question arises as to how this information can be securely integrated in current architectures. It is infeasible to manually collect all the information needed to perform the different privacy checks. Although this might be possible in static (or slowly evolving) systems, it is certainly a daunting and error-prone task. This is overly true if services are aggregated dynamically and used only once. To this end, it is necessary to provide this information as a part of the service description. Together with the syntactic description of the services (e.g. given in WSDL), we provide these meta-data in an adequate XML syntax. This allows for an automated evaluation of meta-data without manual interaction from the users.

4 Discussion: System Requirements

No full control can realistically be performed to verify that the advertised privacy behavior of the services is enforced. However, to provide no control at all is not really a deterrent for dishonest service providers. Technical measures are necessary to increase the degree of trust a user can put in the system and to make unfair data handling more difficult.

The following assertions are made: two technical implementations of services exist: code bundles and Remote (Web) Services. Firstly, no control is possible on remote services. Service Providers have to be trusted to behave as they claim to. External code audit and legal obligations are the main incentives for correct behavior. Secondly, the locally executed bundles can be checked and controlled, as far as suitable technologies are available and compatible with the resources of the users' terminals.

4.1 Requirements of the Remote Service Implementation

The reliability of remote service implementations is a very complex task to tackle. Although different approaches to express the requirements a service complies with exist (such as EPAL [1] or P3P[2]), it is still an open question as

[1] http://www.zurich.ibm.com/security/enterprise-privacy/epal/

[2] http://www.w3.org/P3P/

to how this compliance can be enforced or at least be observed. To this end, we propose a *transparency* approach implemented with the formalization: the users are able to realize what profiles the different service providers might aggregate over time. This will help them understand where privacy issues might arise and which transactions they perform only with providers they really trust or which transactions do not pose large privacy issues.

It is obvious that meta-data provided with the services must be correct. The compliance of the meta-data with the actual data transfered can be observed by technically interested users. This means that ways to observe the correctness exist. Although they might not be accessible for everyone, due to a lack of technical knowledge, this can be seen as a trust building mechanism. On the one hand, users can trust that someone with a better technical background will find the flaws, on the other hand, caring users might ask a trusted third party to check the meta-data for compliance. So the requirements for remote service implementations are rather low, if we are to implement the observability of privacy issues only. If we also wish to take into account the enforcement part, this is still an open research question. For an overview of the issues that arise, see the excellent classification and discussion in [3].

4.2 Requirements of the User Platform

The requirements of the OSGi user platform are as follows. First, the platform must prevent actions that are not compatible with the defined policy. Secondly, services need to be isolated from each other. Thirdly, local sniffers should not be able to interact with trusted services, even when they are installed on the user platform.

Unauthorized action prevention can be performed through Java Permissions. For instance, network access or file system access can be forbidden to specific bundles. The limitation of Java Permissions is that they must be set at launch time of the virtual machine and cannot be dynamically modified. Since the OSGi platform support the addition and removal of code bundles without system restart, occasional reboot cannot be an option of updating the permission configuration. Permissions can be set according to the code location or according to the signer of the code. Since code location is dynamic in OSGi based systems, digital code signature should be used to bind bundles with suitable permissions. This means that permission profiles must be defined beforehand and that the Service Provider signs each bundle with the signature for this permission profile. A risk exist that a given bundle does not exactly match a defined profile and must thus be granted more permissions than it actually needs.

The second privacy requirement is that services cannot access each other without control. Controlling whether a given bundle can provide or look up services is a default behavior of the OSGi Platform [8]. However, no mechanism currently exist to control the services a bundle (or the services it provides) can access. This clearly makes the enforcement of the policies defined in section 3.2 currently not possible. A radical solution is to provide strong namespace isolation between the services, as proposed in the Virtual OSGi approach [13]. However, no fine-grained control is possible, since all interactions between services are prevented. Therefore, the approach is used to support multi-provider environments. For fine grained isolation, additional mechanisms are required. They are defined in section 4.3.

The last privacy requirement is the protection against local sniffers: Unidentified code should not be able to access services that are regularly installed. Such alien code should be rejected at install time through the security mechanisms (see section 2). If they are not, fine-grained service isolation mechanisms should prevent them from accessing other resources.

4.3 Isolation between Bundles

To support privacy policies, a fine-grained isolation mechanism must be defined. Two possible approaches exist. First, the OSGi platform is to be modified to control the service resolution and to check its compatibility with the defined policy. Secondly, it is possible to adapt the existing Service Binding mechanisms to provide this isolation without modifying the framework itself.

The mechanism that supports service publication and resolution in the OSGi framework is the Bundle Context. This context must therefore be modified so as to support isolation between the advertised services. The Bundle Context is accessed through an inversion of control mechanism, *i.e.* each bundle has a reference to the Context and uses this reference to perform service management operations. Consequently, when the Context performs a service lookup, it cannot easily identify the bundle (or the service) on behalf of which the lookup is performed. The Context itself needs to be modified: a bundle should be able only to look up services it is allowed to access. This can be achieved through a RestrictedContext, which is generated at the installation of the bundle and contains a list of the services that can be resolved at lookup. When the bundle is started through its Activator class, it gets a RestrictedContext, instead of a BundleContext instance. Inheritance is exploited to preserve transparency for the bundle code. The registerService, getServiceReference and getServiceReferences methods are overwritten to enforce the policy. Since a specific RestrictedContext instance exist for each bundle, policy runtime update is possible. For the same reason, all services provided by a given bundle must share the same privacy profile.

An alternative to BundleContext modification is the definition of a Secure Service Binding, which would be based on the OSGi service binder [2]. A third party bundle is used to perform the linking between the client service and the 'servant', *i.e.* the service that is advertised. The privacy policy could easily be enforced in the Service Binder. Moreover, different implementations of the Service Binder could provide different types of access control policies: Role Based Access Control, Mandatory Access Control, Discretionary Access Control, and so on. However, there is an important limitation to this approach: the service binder is an optional layer to OSGi and making it mandatory would prevent all bundles that are not specifically developed for it to be used (at least without modification).

5 Conclusion and Perspectives

We have presented a security architecture, formalization and classification of meta-data for hybrid service systems. We could furthermore show that different requirements for remote services and downloadable bundles exist. To this end, it is necessary to find integrated approaches that can cope with the issues related to remote services as well as with those related to downloadable bundles.

These results form the basis for a closer investigation of the complex interplay of static security requirements and dynamic ones arising from the dynamic aggregation of code from different sources and the usage of remote services.

References

[1] A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum. Privacy and contextual integrity: Framework and applications. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 184–198, Washington, DC, USA, 2006. IEEE Computer Society.

[2] H. Cervantes and R. S. Hall. Automating service dependency management. in a service-oriented component model. In *ICSE CBSE*, 2003.

[3] M. Hilty, D. Basin, and A. Pretschner. On obligations. In *10th European Symposium On Research In Computer Security*, LNCS. Springer, September 2005.

[4] S. Höhn. Bringing the user back into control: A new paradigm for usability in highly dynamic systems. In *Trust and Privacy in Digital Business*, Lecture Notes in Computer Science, pages 114–122. Springer Verlag, 2006.

[5] S. Merk, K. Scheidemann, M. Rudorfer, T. Stauner, J. Gruenbauer, G. Popp, and G. Wimmel. Security for downloadable automotive services. In *ESCAR Conference*, 2004.

[6] G. Müller. Privacy and security in highly dynamic systems: Introduction. *Commun. ACM*, 49(9):28–31, 2006.

[7] G. Müller and A. Pfitzmann, editors. *Multilateral Security in Communications*. Informationsecurity. Addison-Wesley, 3. edition, 1999.

[8] OSGi Alliance. Osgi service platform, core specification release 4. Draft, July 2005.

[9] P. Parrend and S. Frenot. A security analysis for home gateway architectures. In *International Conference on Cryptography, Coding & Information Security, CCIS 2006, November 24-26, Venice, Italy*, November 2006.

[10] Rafael Accorsi. On the Relationship of Privacy and Secure Remote Logging in Dynamic Systems. In *Proceedings of the International Information Security Conference Security and Privacy in Dynamic Environments*, 2006.

[11] Rafael Accorsi and Adolf Hohl. Delegating Secure Logging in Pervasive Computing Systems. In *Proceedings of the Internation Conference on Pervasive Computing*, 2006.

[12] J. Raue. B2v palace – presence and location awareness in a collaborative environment. Technical report, BMW Research and Albert-Ludwig-University Freiburg, 2007.

[13] Y. Royon, S. Frenot, and F. LeMouel. Virtualization of service gateways in multi-provider environment. In *Component Based Software Engineering*, 2006.

[14] S. Sackmann, J. Strüker, and R. Accorsi. Personalization in privacy-aware highly dynamic systems. *Commun. ACM*, 49(9):32–38, 2006.

[15] S. Wohlgemuth and G. Müller. Privacy with delegation of rights by identity management. In G. Müller, editor, *ETRICS 2006*, volume 3995 of *Lecture Notes in Computer Science*, pages 175–190. Springer Verlag, 2006.