Privacy Protection for Smartphones: An Ontology-Based Firewall

Johann Vincent, Christine Porquet, Maroua Borsali, and Harold Leboulanger

GREYC Laboratory, ENSICAEN - CNRS
University of Caen-Basse-Normandie,
14000 Caen, France
{johann.vincent,christine.porquet}@greyc.ensicaen.fr,
{maroua.borsali,harold.leboulanger}@ecole.ensicaen.fr

Abstract. With the outbreak of applications for smartphones, attempts to collect personal data without their user's consent are multiplying and the protection of users privacy has become a major issue. In this paper, an approach based on semantic web languages (OWL and SWRL) and tools (DL reasoners and ontology APIs) is described. The proposed semantic firewall takes its decisions (authorize or forbid some action) on the basis of a set of privacy protection rules grounded on two ontologies respectively modeling identity of mobile phone's users and privacy policies. To validate this ontology-based approach, a proof of concept involving a real privacy threat scenario is implemented in Java and the porting of the semantic firewall to the Android platform is outlined.

Keywords: Privacy protection, ontologies, smartphones, semantic firewall.

1 Introduction

In the past few years, the mobile market has rapidly evolved from feature phones to smartphones [1]. It is assumed that the smartphone market will continue to grow in the upcoming years [2]. That evolution is impacting the mobile application market. In particular, the distribution model is progressively switching from a market controlled by telecom operators to online markets such as the App Store or the Android Market. The result of this opening is the recent boom in the number of mobile applications. Since many of these applications are collecting personal data with or without the consent of the user [3], the issue of an enhanced protection of the user's privacy must be addressed.

In this paper, we claim that an ontology-based firewall can effectively protect the user's digital identity and personal data. Ontologies provide a shared vocabulary, which can be used to model a domain, that is, the type of objects and/or concepts that exist, together with their properties and relations [4]. Thanks to an explicit knowledge representation of the data requested by any mobile application, the firewall can determine whether the application requests are permitted or forbidden, according to predefined customized security policies.

C.A. Ardagna and J. Zhou (Eds.): WISTP 2011, LNCS 6633, pp. 371–380, 2011.
© IFIP International Federation for Information Processing 2011

This paper is organized as follows. In section 2, several ontologies for digital identity and privacy protection mechanisms are reviewed and discussed. In section 3, our approach to achieve privacy protection for smartphones is detailed: the global architecture of the firewall is described, as well as the two distinct ontologies that have been specified and implemented in OWL language [5]: one dealing with digital identity and the other with privacy concerns. In order to explain how the proposed firewall responds to a common privacy threat, a basic scenario using policy rules expressed in SWRL [6] (Semantic Web Rule Language) language is explained step by step and the porting of the semantic firewall to the Android platform is oulined. Conclusions and future work are given in section 4.

2 Related Works

2.1 Protection on Smartphones

To protect their operating systems from malicious software, operating system developers have implemented various protection mechanisms. On iOS or BlackBerry OS for instance, applications are made available to customers after going through an agrement process that verifies that they do not contain unwanted code. Android also encourages developers to sign their applications with a trusted certificate but it is not mandatory.

In addition to this signature mechanism, Android and BlackBerry prompt the user with a manifest during the installation process. This manifest shows the permissions granted to the application and the user must accept them in order to install the application. The problem with that kind of protection is that users tend to accept the manifest without really assessing all its consequences. The BlackBerry OS tries to address this issue by allowing a modification of the permissions for each application outside the installation process.

However, to our knowledge, there is no real-time privacy protection mechanisms implemented on current platforms that can prevent an application to access specific data.

2.2 Identity Ontologies

Before building a semantic firewall that can efficiently protect users from privacy breaches, an exhaustive record of all the data that need to be protected must be done. Thanks to their declarative form, ontologies are the best way to explicitly represent the manifold digital identity of users that are juggling daily with several avatars, nicknames, passwords, telephone numbers, email accounts, homepages and so on.

On social networks, countless people are describing themselves and part of their private life on their home page. FOAF (Friend Of A Friend) uses W3C's RDF technology to represent such information as an ontology [7]. The core of FOAF describes characteristics of people and social groups, the networking being achieved thanks to the *foaf:knows* property. In addition to the FOAF core

terms, one can also describe Internet accounts, mailboxes, homepages etc. This general-purpose ontology is well suited for social network identities but it lacks information regarding mobile phone identities.

Some of this missing information can be found in the vCard file standard format for electronic business cards. vCards contain the user's personal and professional affiliation, address and geolocalisation, email, URLs, photos, logos and even audio clips. They can be attached to email messages, directly embedded in web pages (hCard microformat for (X)HTML) or represented into XML/RDF. The corresponding ontology can be found in [8]. Thanks to that ontology, it is possible to specify fine grained information, for instance indicating that a phone number is also a fax number or telling which email address should be given preference to others.

2.3 Privacy Protection Ontology-Based Policies

A policy is an enforceable, well-specified constraint on the performance of a machine-executable action by a subject in a given situation. Web semantic languages are particularly suited for representing, reasoning and enforcing policies. Thanks to policies, it is possible to adapt the behaviour of a complex system without changing pieces of code. In [9], three approaches are compared and discussed: Ponder, Rei and KaoS. Only the last two are ontology-based and are both written in OWL language.

Rei [10] proposes an application-independent ontology to represent the concepts of rights, prohibitions, obligations and policy rules. It also includes a general class describing the action to be performed, together with preconditions, target objects and results. KaoS has been developed within the broad context of multiagent and distributed systems. It is a complete framework for domain and policy services. Among KaoS features, there is a GUI called KPAT (KAoS Policy Administration Tool) allowing people to manually specify, analyze, and modify authorization and obligation policies, thus hiding the complexities of OWL from end-users. Policy decisions are performed by so-called *Guards* that store precompiled policies and maintain a history of actions. KAoS framework is a very rich environment. However, a simplified framework without the multiagent paraphernalia would better suit our needs.

3 Our Approach

Our objective is to create a semantic firewall between the applications and the private data of the smartphone owner. We use ontologies to represent both the concepts of identity and personal data and also to model privacy policies. The proposed firewall relies on those ontologies to block or authorize a request. A request consists in some actions performed by an agent on another agent data. In our model, the applications are issued by a service provider and we consider that the data requests are made on its behalf.

3.1 Architecture

The global architecture (Figure 1) is grounded on a smartphone ontology written in OWL that includes two ontologies: the ontology designed to represent privacy policies and the ontology of the digital identity stored in the mobile. The firewall is in charge of populating the smartphone ontology with the individuals corresponding to the specific request. When a request is made, the firewall processes the request by calling the description logic reasoner (DL reasoner) in charge of inferences. These inferences are performed according to a set of policy rules written in SWRL. Our algorithm is inspired by the SOUPA algorithm [11]; three cases can occur after classification:

1. There is no policy to manage the request. The firewall then applies the default rule, which can be either liberal (all actions permitted) or conservative (all actions forbidden).
2. Two or more policies are in conflict. As in the first case, the default rule is applied.
3. An adapted policy exists. It is applied and the action is classified either as permitted of forbidden.

The interactions between the service provider and the firewall are described on the sequence diagram of figure 2. The firewall first identifies the service provider that makes the request. The firewall successively registers the service provider and the request by creating them as individuals in the ontology. Then, the DL reasoner is launched to classify the concepts of the ontology and decide the type of action (permitted/forbidden). Finally, if the action is permitted, the firewall provides the appropriate data and logs the request in the ontology for history purposes.

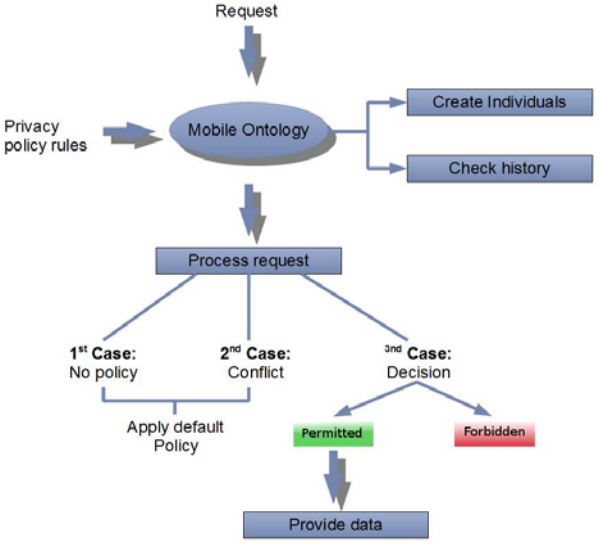

Fig. 1. Flowchart of the semantic firewall

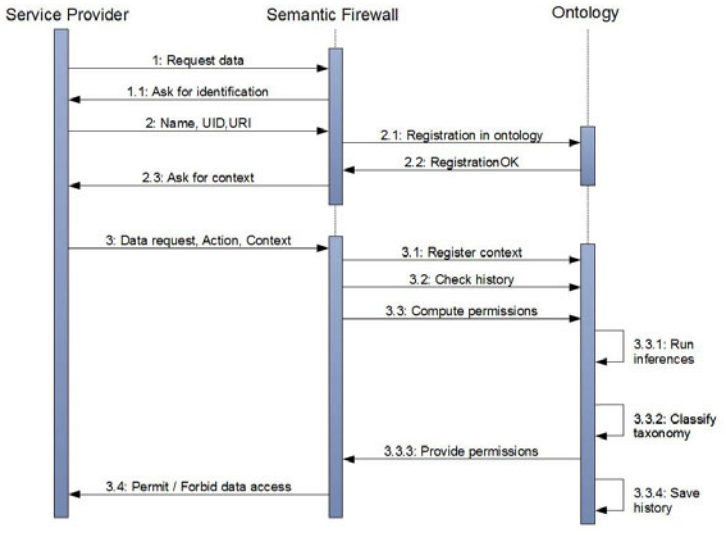

Fig. 2. Sequence diagram of the semantic firewall

3.2 Ontology of Identity on Smartphone

The ontology of identity on smartphone describes the concepts on which the privacy rules are applied. We have defined the following classes and properties:

- name, first name, date of birth and place of birth;
- contact information: postal address, phone number, email address;
- location data: GPS, IP address, Cell Id;
- IMSI: International Mobile Subscriber Identity;
- IMEI: International Mobile Equipment Identity;
- directory of contacts;
- certificates and cryptographic keys.

These concepts are organized under three main classes: the *Agent* that can either be an individual or an organization, the agent's *IdentityInformation* and the agent's *Data*. The building of this ontology was done with the Protégé-OWL editor [12]. We also used Protégé to populate the ontology with individuals such as telecom operators and personal data.

3.3 Lightweight Ontology for Privacy Policy

To model privacy policies applied to smartphones, a lightweight ontology (figure 4) was built with Protégé. The two main classes of this ontology are *Policy* and *Action*. A policy controls an action, is created by an agent and has a date of creation. An action has the following attributes: date of the action, actor, context in which the action is performed and finally the data on which it is performed.

These information are used to manage the history of the request. As previously mentioned in the architecture description, that ontology is the key element for the classification of action in two categories: permitted or forbidden. We used the Pellet DL reasoner [13] plugin for Protégé to check that this classification is accurate. Due to the fact that the two classes *PermittedAction* and *ForbiddenAction* are not disjoint, in case of conflict, the action will be classified under both. That is how our firewall detects conflicts.

3.4 Rules for Privacy Protection

The firewall is based on the application of privacy rules defined in SWRL. This rule format was chosen because it is both supported by Protégé and the DL reasoner. The rule base is stored appart from the ontology. Rules are separated from the ontology in order to make the addition of new rules and maintenance easier. Listed below are some rules in the case of a liberal default policy which explains why they all are forbidding rules.

Rule 1: Forbid any action made by a service provider with an invalid certificate.

```
[policy1: (?s rdf:type id:ServiceProvider) , (?a rdf:type id:Action),
 (?c rdf:type id:InvalidCertificate), (?s id:hasCertificate ?c)
 -> (id:policy1 id:forbids ?a)]
```

Rule 2: Forbid access to location data if the service provider is SP1.

```
[policy2: (?a rdf:type id:Action), (?a id:hasActor id:SP1),
 (?d rdfs:domain id:LocationData), (?a id:hasTarget ?d)
 -> (id:policy2 id: forbids?a)]
```

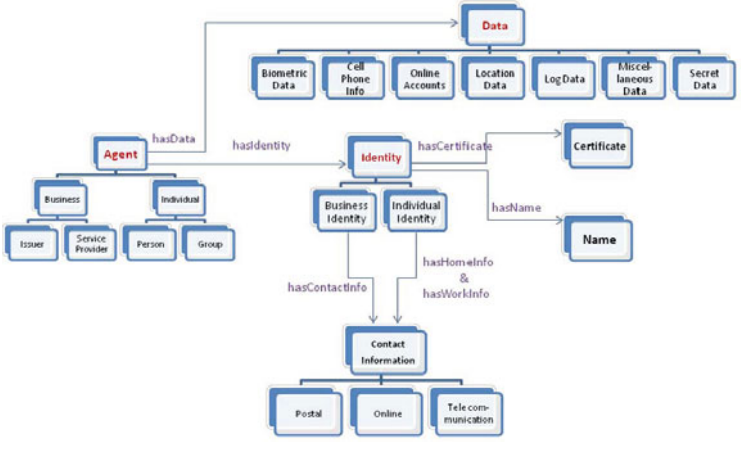

Fig. 3. Digital identity ontology for smartphones

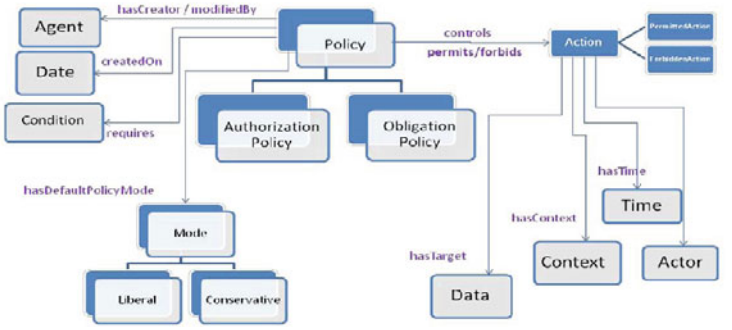

Fig. 4. Privacy ontology for smartphones

Rule 3: Prevent a service provider form obtaining postal address if it has already gathered the phone number.

```
[policy3: (?a rdf:type id:Action), (?c rdf:type id:ServiceProvider),
 (?a id:hasActor ?c), (?d rdfs:domain id:Postal), (?a id:hasTarget ?d),
 (?c id:hasHistoricTarget id:phoneNumber1) -> (id:policy3 id: forbids ?a)]
```

3.5 Validation

Privacy threat scenario

To demonstrate the interest of an ontology-based firewall, we devised a privacy threat scenario in which we want to prevent the collection of location data (GPS, IP address and wireless information) from a service provider (SP2). Our experimentation was conducted in three steps. First, the SWRL rules corresponding to our scenario were written. In our case, only one rule is required, since the IP address, wireless information or GPS are all subclasses of *LocationData*:

```
[policy5: (?a rdf:type id:Action) (?a id:hasActor id:SP2)
 (?d rdfs:domain id:LocationData) (?a id:hasTarget ?d)
 -> (id:policy5 id:forbids ?a)
```

Then, all the individuals needed by the scenario were created under the Protégé editor: SP2 as an instance of *ServiceProvider*, the user as an instance of *Person* with all his location data and SP2's request under the *Action* class. The rule base was launched within Protégé and policy rule number 5 was triggered. The DL reasoner was then executed to classify individuals. As explained before, the classification under the *ForbiddenAction* class confirmed that SP2 was not allowed to collect location data.

Proof of concept

A proof of concept of our firewall was implemented as a client/server application. We chose the client/server approach because we wanted to mimic a real

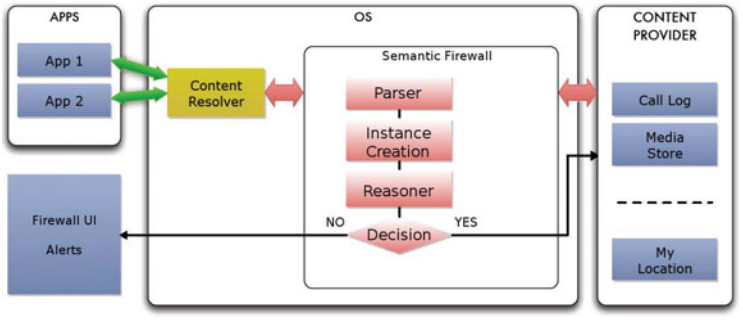

Fig. 5. Android porting of the semantic firewall

access scenario where the application (client) requests the data through a firewall (server). Both the client and the server are developed in Java since it is widespread for mobile phone programming. The Java language is also interesting because it supports specific API, such as JENA [14], for ontology interactions. The server was developed to be as generic as possible while the client was designed to match our test case. However, a graphical user interface was added on the client side to allow the tuning of the request parameters.

Our tests were done on a standard PC and they validate our approach by successfully preventing the access to any location data. The two ontologies (94ko) contain around fifty classes and a hundred object/data properties. In our tests, with five policy rules and around twenty individuals, the execution time is less than 100ms which is reasonable for a mobile usage.

Smartphone architecture

The next step of our work was to implement the semantic firewall on an actual smartphone based on the Android OS. Due to the specific nature of the Android smartphone platform, our client/server application had to be adapted. The major modifications to our firewall are described in the next paragraphs.

First of all, we chose the Android platform as it is free and allows modifications of the operating system. In fact, the operating system is the mandatory entry point of any data request made by any application. It thus appeared to us as the perfect place to implement our semantic firewall. Moreover, Android is also written in JAVA and a JENA API porting is available: AndroJENA [15]. Finally, the way to exchange data is centralized on Android as every request has to go through a single class: *ContentResolver*.

The major modification we had to make to our initial model was to review the way an application requests data. In our PC proof of concept, we made the simplistic assumption that the requests made on the client-side could be processed on-the-fly by the semantic firewall. It is unrealistic to hope that application developers will follow such an approach. To address this issue, a parser was added to the semantic firewall. It is charge of extracting from the request an <actor, action> pair and of providing a default context of execution. The actor, action

and context of execution are then reified as instances of our ontology and fed to the DL reasoner. The new firewall complying with the Android specifications is described on Fig. 5.

4 Conclusions and Future Work

As semantic web techniques, languages and tools are coming of age, developing ontology-based applications is getting more straightforward. This paper demonstrates that it is possible to build a convincing prototype of semantic firewall for smartphones. The proposed firewall answers a growing concern: data collection without the user's explicit consent. It takes advantage of the expressiveness of OWL. Thanks to the Open Data Movement [16], more and more ontologies are available and can be reused and adapted to fit specific needs. Our application makes the most of the powerfulness of DL reasoners: all useful inferences are made by the classification mechanism. Policy rules are defined declaratively and separately from the ontologies. Thus the addition of new rules is made easier. The firewall is implemented in Java, using the Jena ontology API and response times to requests are less than what was expected, the call to the DL reasoner, being the most time-consuming operation.

Future works will mainly focus on extensive unit testing of our Android porting. Only once the functioning of the semantic firewall is satisfactory can we focus on the delicate issue of the storage of the ontology. Since smartphones are coupled with an embedded SIM card [17], one has to decide whether the whole ontology, or only part of it, can be stored directly on the SIM card. In particular, we think that the Smart Card Web Server technology [18] can be a promising solution for allowing some part of the handset operating system to be interfaced with the embedded SIM card ontology. The use of SWRL to write policy rules may also change, since RIF (Rule Interchange Format) has recently become a W3C Recommendation (june 2010) [19].

Acknowledgements

The authors gratefully acknowledge the sponsorship by SFR, the French second-largest wireless telecom operator. We especially want to thank Jean-Philippe Wary, initiator of the project at SFR.

References

1. Gartner. Gartner says worldwide mobile phone sales grew 17 per cent in first quarter 2010 (May 2010)
2. Gartner. Android to become no. 2 worldwide mobile operating system in 2010 and challenge symbian for no. 1 position by 2014 (September 2010)
3. Lookout. The app genome project (July 2010), `http://blog.mylookout.com/2010/07/introducing-the-app-genome-project/`

4. Allemang, D., Hendler, J.: Semantic web for the working ontologist. In: Effective Modeling in RDFS and OWL. Morgan Kaufmann, San Francisco (2008)
5. McGuinness, D.L., Van Harmelen, F., et al. Owl web ontology language overview. W3C recommendation, 10:2004–03 (2004)
6. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: Swrl: A semantic web rule language combining owl and ruleml. W3C Member submission, 21 (2004)
7. Brickley, D., Miller, L.: Foaf vocabulary specification 0.98. Namespace document, FOAF Project (August 2010), http://xmlns.com/foaf/spec/20100809.html
8. Iannella, R.: Representing vcard objects in rdf/xml, 12 (2001), http://www.w3.org/Submission/vcard-rdf/
9. Tonti, G., Bradshaw, J.M., Jeffers, R., Montanari, R., Suri, N., Uszok, A.: Semantic web languages for policy representation and reasoning: A comparison of kAoS, rei, and ponder. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 419–437. Springer, Heidelberg (2003)
10. Kagal, L., Paolucci, M., Srinivasan, N., Denker, G., Finin, T., Sycara, K.: Authorization and privacy for semantic web services. IEEE Intelligent Systems, 50–56 (2004)
11. Chen, H., Finin, T., Joshi, A.: The soupa ontology for pervasive computing. In: Ontologies for agents: Theory and experiences, pp. 233–258 (2005)
12. What is protégé-owl? http://protege.stanford.edu/overview/protege-owl.html
13. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl reasoner. Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 51–53 (2007)
14. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: implementing the semantic web recommendations, pp. 74–83 (2004)
15. Androjena. Jena android porting (2010), http://code.google.com/p/androjena/
16. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009)
17. TS ETSI. 102 221:" uicc-terminal interface: Physical and logical characteristics". ETSI Standard (2010)
18. Urien, P.: Internet card, a smart card as a true internet node. Computer Communications 23(17), 1655–1666 (2000)
19. Kifer, M.: Rif overview. W3C Working Group Note (2010)