An Approach for Realizing Privacy-Preserving Web-Based Services*

Wei Xu R. Sekar I.V. Ramakrishnan
Department of Computer Science
Stony Brook University
{weixu, sekar, ram}@cs.sunysb.edu

V.N. Venkatakrishnan
Department of Computer Science
University of Illinois at Chicago
venkat@cs.uic.edu

Categories and Subject Descriptors:
D.2.4 [**Software Engineering**]: Software/Program Verification

General Terms: Security

Keywords: Privacy, web service, information flow

1. INTRODUCTION

The past decade has witnessed a phenomenal growth in the number of users who routinely use the web to obtain information, conduct research, or carry out financial transactions. While the ability of the web to provide customized information and financial services has boosted personal and business productivity, it has raised significant concerns regarding an individual's privacy. The privacy concerns are exacerbated further with the advent of more complex services that involve multiple service providers, as it raises the potential of personal information being shared across these providers in ways that weren't intended by the owner of the information.

To illustrate the increased privacy concerns raised by composite web-based services, consider a web-based service that allows an individual to research and order over-the-counter medication (or health supplements) at a low cost, while ensuring that there are no adverse interactions based on the individuals current prescription. Such a service will require access to the individual's personal medical history, and will also require interaction with other entities such as web-based medical references, discount pharmacies and credit-card companies. Users may not want their personal health histories to be shared with discount pharmacies or credit card companies. Similarly, they may not want their credit card information to be shared with providers of free on-line information.

Web-based service providers have long recognized the importance of addressing user privacy concerns, and taken steps to state their policies regarding the storage and use of personal user information. Unfortunately, since these policies are stated in English, they are imprecise and ambiguous, posing problems in accountability and enforceability, as well as in checking compatibility between consumer and provider policies. Indeed, the verbose nature of these policies discourage most users from even reading them!

To address the above problem, we present a new approach where the consumers as well as providers can express their privacy concerns in a *formal way*. Specifically, consumers express their requirements in the form of *policies*, while providers specify their use of consumer data using *models*. Our approach automates compatibility checking between policies and models. If there is an incompatibility, the consumer is informed how she can refine her policies in order to use the service. If she does not want to change her policies in any way, the approach passes on additional privacy requirements to the provider. Service access can continue in case of incompatibilities only if the consumer relaxes her policies, or the provider honors additional consumer privacy requirements.

The key idea behind our approach is a judicial combination of *trust* (on service providers to accurately specify use of consumer data) and *verification* (for compatibility resolution). This combination enables our approach to support privacy preservation *without requiring access to proprietary code* that implements the service. This contrasts with previous approaches for privacy policy enforcement in software [2], which relied on source code analysis.

We point out that in general, *there is no technological solution to the problem of malicious providers.* Even source-code analysis techniques are of no avail here: these techniques can ensure that a *given piece of software* satisfies a certain policy, but in a distributed environment, consumers cannot prevent a malicious service provider from simply switching to a different piece of software that violates these policies. As such, technology can't prevent malicious providers that misrepresent their terms of usage, or provide incomplete information about it. Indeed, our approach presupposes that due to market or societal forces, providers and consumers have already decided to collaborate in order to preserve consumer privacy, and that our approach simply provides a technological basis to facilitate this collaboration.

2. PRIVACY-PRESERVING FRAMEWORK

As mentioned earlier, we are primarily interested in privacy preservation in the context of complex services that involve multiple service providers. We envision that such services are realized by composing *component services*, each of which may be provided by a different provider. To illustrate service composition, consider a travel management service, which is composed from an air ticket booking service (Service A), a hotel reservation service (Hotels.com), a rental car reservation service (Service B), and a driving direction service (MapQuest). Some of these services may in turn use other services, e.g., Service A may obtain its services from JetBlue, Southwest and Orbitz, while Service B may use Avis, Hertz, and Dollar.

Users can use the composite service as a one-stop shop for their travel planning and reservations. To use the service, a user first provides his travel information (such as the origin/destination city) and payment information (such as the credit card number) to the composite service. The composite service will invoke Service A with the origin/destination and payment information to book an air ticket, request Hotels.com to reserve a hotel near the destination address, invoke Service B to reserve a rental car, and use MapQuest to get driving directions from the airport to the hotel.

To protect users' privacy in the composite service, the user needs first define a privacy policy, e.g., travel origin/destination and payment information are allowed to be sent to JetBlue, Orbitz, Hotels.com, Avis and Hertz, but only the destination address is allowed to be sent to MapQuest and Dollar. Each component service is required to provide a model that describes how its input data are handled to different principals. For instance, because the driving di-

*This research is supported by a grant from Computer Associates and an ONR grant N000140110967.

WWW 2005, May 10–14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

rection service is very simple, its model may be described as "destination airport and hotel information are sent to MapQuest." The composition code uses the models to verify if the services violate the given privacy policy before executing these services. Because data can flow from one service to another service in the composition code, we use a program transformation technique to instrument the composition code to track such information flows as well as perform policy verification checks.

Three important components are introduced in the framework for the purpose of preserving users' privacy in composite services: *privacy polices*, *service models*, and *policy enforcement*. (A more detailed description on the framework is provided in [3].)

Privacy Policies. In our approach, consumers can express their privacy concerns using *privacy policies*. These policies capture what types of consumer data they are willing to share with which types of service providers ("*principals*.") The types of consumer data are identified using *data labels*. For instance, a credit-card number may be classified as "highly sensitive," a telephone number as "sensitive," and the country of residence as "public." The types of service providers are represented by *principal labels*. For example, JetBlue, Southwest, Orbitz, Avis and Hertz may be classified as "trusted," and MapQuest and Dollar as "untrusted." It is possible that multiple labels may be associated with the same data or service provider, each capturing one aspect about its use. For instance, data regarding a consumer's prescriptions may be labelled as "health" and credit card information may be labelled as "financial," while both of them may be classified as "highly sensitive."

Policies simply state that data with a certain data label can be made available only to principals with certain labels. For example, a user may have a privacy policy specifying that "*address and financial data can be sent only to trusted principals*."

The set of labels in a privacy policy form a lattice, with a partial order based on either the natural ordering relationship among the labels (e.g., *sensitive* $\leq$ *public*) or the subset relationship or a combination of both. This lattice is used in policy enforcement checks. Higher level abstractions to further simplify policy definition can be layered on this simple policy language, but this aspect is not discussed further in this paper.

Service Models. When a consumer provides data to a service provider, he wants to ensure that it is used in a manner consistent with his policy. To verify if this will be the case, the consumer requests a *model* of the service, which captures the manner in which the service uses consumer data. The model captures how the information provided to the service can flow to various principals. For instance, the model of the air ticket booking service might be described as "origin/destination cities and credit card information are sent to JetBlue, SouthWest and Orbitz." If the service does not use other services in turn, then there is only one principal involved, namely the service provider itself. If the provider uses other services, the providers of these services will also need to be represented in the model.

A model can use *wildcards* for principal names, e.g., the rental car reservation service may state that credit card information may be sent to unspecified principals. Wildcards may be used either if the provider considers the principal information to be proprietary, or if there are far too many principals to list them all in a model.

To facilitate privacy preservation in the context of composite services where data can flow from one service to another service in the composition code, each service model is also required to describe how the input information to the service can flow to the outputs of the service. Constraints, such as *exact dependence* or *quantitative dependence*, can be defined on these flows to specify how much information contained in the inputs is flowed to the outputs. The composition code will make use of these flows and their associated constraints to propagate data labels from one service to another for policy compliance checks.

Policy Enforcement. When a consumer receives the model, she can check if the model is compatible with her policies. To do this, the consumer labels each of the data items that she is providing as input to the service and labels every principal mentioned in the service model. (In reality, labels will, in most cases, be generated automatically by a user application). In addition, she provides her privacy policy. To perform policy compliance check, the consumer's application uses the model to discover the labels on the data that will be sent to different principals mentioned in the model. Then it checks if the consumer's policy allows data with these labels to be sent to those principals. If the check succeeds, the application forwards the request to the service provider. If the policy is violated, there are several possible solutions:

- relax privacy policy so that the service can be used. In this case, the user is told the manner in which the policy is being violated, e.g., credit card information is being sent by Service B to unspecified principals; or
- forward additional privacy obligations to providers, e.g., specify to service B that it is authorized to share the credit card information only with Hertz and Avis; or
- look for an alternate service provider that satisfies user policies.

The composition code is transformed to add code for data label tracking and privacy policy enforcement. The basic idea of the transformation is to introduce an auxiliary variable for each program variable, which holds the data labels of the program variable. These data label variables are maintained at runtime along with updates to program variables and are used for policy enforcement.

3. IMPLEMENTATION CONTEXT

To illustrate the practicality and utility of our framework, we have built several demonstrative composite web services (e.g. an electronics comparison shopping service, a travel management service) in which the preserving of privacy is based on our framework.

Our implementation is based on *Personalized Information Agents (PIAs)* [1], which are intelligent agents that are capable of navigating web sites to access the capabilities provided by the web site without requiring explicit navigation actions by the user. They provide a programmatic interface to access these capabilities. Thus, PIAs can be thought of as a "wrapper" that layers over web sites that are intended for use by humans, and in effect, turn them into web services. This factor enables us to proceed with the implementation without having to be constrained by the limited deployment of *open* web services.

4. REFERENCES

[1] A. Gandhre, P. Santhanagopalan, P. Singh, D. Ramavat, I. Ramakrishnan, and H. Davulcu. Creating and managing personal information assistants via a web browser: The WinAgent experience. In *Workshop on Information Integration on the Web*, Toronto, August 2004.

[2] A. Sabelfeld and A. C. Myers. Language-based information-flow security. *IEEE J. Selected Areas in Communications*, 21(1), Jan. 2003.

[3] W. Xu and R. Sekar. A practical approach for privacy-preserving web-based services. Technical Report SBU-CS-SL-05-1, Stony Brook University, Report produced for Computer Associates, January 2005.