A User-Centric Authentication and Privacy Control Mechanism for User Model Interoperability in Social Networking Sites

Yuan Wang and Julita Vassileva,

Department of Computer Science
University Of Saskatchewan Saskatoon SK,
Canada S7N5C9 1-306-966-4744

yuw193@mail.usask.ca jiv@cs.usask.ca

Abstract. In this paper, we present a new authentication and privacy control mechanism for personalized mashups of social networking sites. Current authentication and privacy control mechanisms lack flexibility and transparency. This mechanism can make the user model interoperation process for mashups more transparent to users. Users can have a clear understanding and control about which part of their data is being accessed by the mashup application. This mechanism is an important part of user model interoperability framework.

Keywords: Personalized mashup, social networking sites, user model interoperation, authentication, privacy

1 Introduction

One of the most important features of Web 2.0 is that it is social: users can share content with their friends and can develop social ties among each other. Social features can be combined with domain-specific applications, e.g. a music application like LastFM, to empower a community of users. Reusing existing user model data from the domain-specific application (e.g. preferences for particular groups or music genres) can minimize the effort for users, allow useful adaptations and recommendations to be provided by other applications, and thus may help bridge the gap across their presence in different communities. Many researchers in the User Modeling field have investigated how to ensure User Model Interoperability (UMI) by exchanging user model data between applications. Web-based APIs and mashups provide an easier way to implement UMI. A mashup is a web- or desktop- application that combines information and/or services from one or more external sources [1]. Social networking site mashup applications combine user social data with some domain-specific application (e.g. music player/recommender, shopping, or mapping application). At the time of writing, there are more than 50,000 facebook mashup applications. There are two mashup application modes. The first one is where the mashup application runs inside the data provider's page within a frame or gadget,

such as facebook application or OpenSoical gadget. This makes it convenient for the user, allowing interaction with several applications within the same website, avoiding duplication of effort when logging in. The second mashup mode is where the mashup application runs on its own web page. In this mode the user may have to log in in more than one application, if user data is shared among them. There are no significant differences between those models from developer perspective. In both cases, the mashup application actually runs on its own server. In first mode, the social network site is simply a proxy. It displays the mashup application's page within its gadget.

2 Current Technologies

A complete UMI framework must have four parts [4]: (I) user data exposing and discovery; (II) user identification mapping (III) authentication and privacy controls, (IV) user data exchange. A personalized mashup application as light-weight mashup application also has these four parts. The following sections will explain each part briefly. This research mainly focuses on the third part: authentication and privacy controls. To clarify the terms, we use "*data provider*" to denote the application or service which publishes an open API to share user model data; we use "mashup application" to denote the application which requests external user model data and uses it to adapt its own service.

(I) *User data exposing and discovery*: During this process, the data provider publishes user data APIs and information about the semantic and syntactic meaning of the data it provides. Currently, the mashup application developer has to discover data providers manually and read their APIs documents. There are some promising techniques to automate this process: SAWSDL (Semantic annotation of WSDL) http://www.w3.org/2002/ws/sawsdl/, SA-REST (Semantic annotation of REpresentational State Transfer) [7], XRDS (extensible resource descriptor sequence) and XRDS-simple (a simple and subset version of XRDS).

(II) *User identification mapping*: In order to use external user data, the mashup application has to know the user's identity in the data provider's system. Currently, the end user has to provide this information manually in the mashup application. However, there are some universal identity management platforms available. OpenID is the most popular one [5]. Open ID is a decentralized, interoperable, extensible platform for user-centric Internet identity management. OpenID provides users with a universal internet identity which can be used for many online applications. Right now, there are dozens of OpenID providers (Google, Yahoo, Flickr, AOL and etc) and users can choose the ones they trust as their identity providers. With a universal identification management, data provider and mashup provider do not need to map user's identity across two systems.

(III) *Authentication and privacy controls*: The user data is behind the lock of username and password [2]. In order to access user model data from a data provider, the mashup application needs to authenticate itself to the data provider. Here, access means read, edit, add or delete operation on user data. Authentication has two parts: first, validating the mashup application's identity, and second, validating whether the mashup application has the right to access user data. Validating the mashup application's identity is a relatively simple task. The current solution is through API

key and secret key. In order to use the data provider's APIs, the mashup application needs to register at the data provider by presenting some basic information. After the registration, the data provider assigns a pair of API key and secret key to the mashup application. The API key and secret key is like username and password for identifying the mashup application to the data provider application.

Validating whether the mashup application has the right to access user data is a more difficult task. The current solution requires sending username and password of the user in the data provider's system. There are two ways to do the authentication. First, the mashup application directly asks the user's name and password of the data provider system and does the authentication itself. This is risky from a user point of view. Alternatively, the mashup application can redirect the user's web page to the data provider. On the data provider's web page, the user is required to login to authenticate him- or herself. After the user is authenticated, the data provider will inform the user that the mashup application is trying to access his or her data and will request permission to allow the mashup application to access the user data. If the user gives permission, the data provider will "callback", i.e. transfer the user's web page back to the mashup application; the data provider will also send a session key to the mashup application (see Fig. 1).

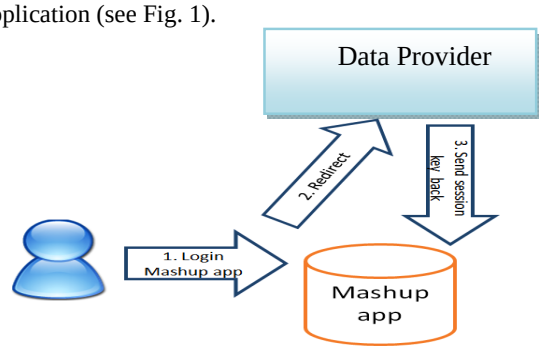

Fig.1. Safe Authentication Model

With this session key, the mashup application can access user data. This session key is used just for one user. Different data providers have their own rules about this "session" authentication. For some data provider, the session key may expire after hours and the user has to authenticate again. For other data providers, the session key may not expire. Some user data require authentication and some do not. This "session" mechanism is inconvenient for users when a mashup invokes several data providers, and needs to do many authentications to many user data providers. Before actually using the mashup application, the user has to authenticate with each data provider, and wait for the page redirecting. This authentication mode is not only inconvenient. It doesn't give users control over the user data interoperation process. Even though only data that is publicly available online is currently shared among applications, privacy concerns have been voiced and users are concerned about having little understanding or control over how data is shared. Users are unable to see which data is shared, how it is used, how long it is kept and have no control other than not adding the third party application (the mashup). The opaqueness of the user model data sharing process often makes users hesitant to use the available services.

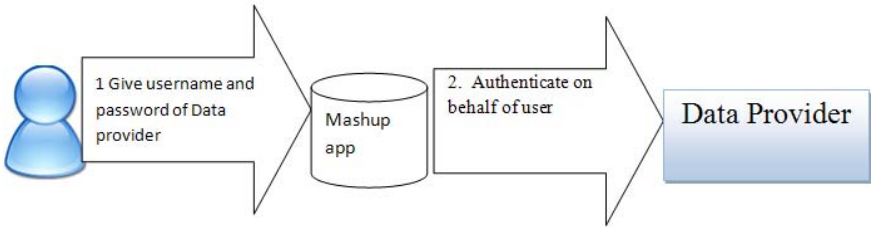

Fig.2 Risky Authentication Model

There are several mature authentication and privacy control frameworks address some of these issues, such as OAuth and Shibboleth. But there are still some limitations of these frameworks, which will be mentioned in section 3 related works.

(IV) *User data exchange*: this process is where user model interoperation really happens. Currently, Web-based API is the major technology used for this task. The most popular protocols of Web-based API are SOAP message and REST. Web-based API is reliable technology. Among the four parts of the user model interoperation, user data exchange is the most mature one.

3 Related Work

Berkovsky et al. [8] pointed out four major challenges for achieving UMI.
1. Systems are unwilling to share user models; 2. Privacy issues;
3. Technical considerations; 4. Semantic heterogeneity among applications

A lot of research has addressed the issue of semantic heterogeneity [4], [8], [9]. This research mainly focuses on the second challenge. There has been also a lot of research on privacy in user modeling [10], [11], [12]. Since the 1980ies, researchers have studied users' attitudes about internet privacy. They found out that users can be divided into three clusters [10], [12]:

1. Privacy fundamentalists, comprising approximately 17% of the entire user pool, generally express extreme concern about any use of their data and an unwillingness to disclose information, even when privacy protection mechanisms would be in place.

2. Pragmatic majority, approximately 56% of the entire user pool, are generally concerned about their privacy as well, but less than the fundamentalists. They are also far more willing to disclose personal information when they are see potential benefits and protection.

3. Privacy unconcerned, who takes 27%, of the entire group, tends to express mild concern for privacy.

In the recent decade, the number of privacy fundamentalists and privacy unconcerned is declining, and there is increase in the number of privacy pragmatic users [10]. In other words, privacy pragmatics is the majority of internet users and

their number is still increasing. Therefore, most internet users care about privacy but are also interested in personalized services. As developers, we need to motivate users to disclose their data and protect their privacy. Previous research, e.g. [10], [12], reports about ways to motivate users to disclose their data. In order to motivate users to disclose their data, the application should tell users the benefits of personalization. Moreover, users want to know how their personal information is being used and to have control over this usage. Applications should be able to explain to users what facts and assumptions about them are being stored, and how these are going to be used. Users should be given ample control over the storage and usage of this data. Trust in a web site is a very important motivational factor for the disclosure of personal information. Trust is built on positive past experience, so applications should allow users to incrementally supply more information as their trust in the application increases.

Therefore, the authentication and privacy control mechanism should make the UMI process transparent to the user. This direction – to give the user control over which partial models should be made available to which applications - was suggested recently by Kay [14]. The user should be aware of the user model data required by a mashup application and the terms of use of the data. Based on this information, the user can decide whether or not to allow the mashup to access and use user data. As mentioned before, there are some frameworks that attempt to achieve that: such as OAuth and Shibboleth. OAuth is an open protocol to allow secure API authorization in a simple and standard method [2], [5]. It is a light-weight framework which has already been adopted by some social networking sites, like Twitter. But OAuth cannot let the user decide how to do the authorization. For example, when the user trusts a mashup application and feels comfortable about letting it access his or her data, the user does not want to be involved in the authorization (it is viewed as an extra burden). The Shibboleth protocol is another mature framework which ensures safe user data sharing between systems [13]. The user can define an attribute release policy to each outside system which requires user data. There are many prerequisites for using Shibboleth: the system must have secure identity management and must install the required software. Shibboleth is ideal for universities and other larger organization.

We propose a new authentication and privacy control mechanism. This mechanism can facilitate privacy control by letting users customize their privacy settings depending on each individual mashup application and their different privacy preferences. Moreover, the user can decide how to do the authorization. This mechanism does not deal with data provider discovery or semantic heterogeneity directly, but it can be integrated with other mechanisms to achieve a complete user model interoperation framework.

4 Authentication and privacy mechanism

As mentioned in the introduction, there are two mashup application modes. In both modes, mashup applications need authentication.

4.1 Application registration

When a mashup application registers at a data provider, the mashup application needs to list all the user data it will access during the service and the type of action on the data: such as read, edit, add, and remove. Besides that, the mashup application also needs to describe the terms of use of the user data, and information about who provides this mashup application. The mashup application is not able to access any user data which are not listed in the registration. The registration information is visible for the user. Therefore, the user knows what kind of data will be used by the service. When the user wants to use a new mashup application, the user data provider will show this application's registration information to the user.

4.2 Authorization

When the user invokes the mashup application for the first time, the mashup application will redirect the user to the data provider. The data provider will ask the user to login. After user login, the data provider will show the registration information about the mashup application (as shown on Fig. 4) and ask the user if he or she wants to authorize the mashup application. The user can grant the mashup application one of three levels of access. The first level is access *without user authentication*, i.e. the mashup application can access the user data it registered without user authenticating. This would be very convenient for the user since it will require no further effort for authentication; however, it gives the mashup application the right to access the user data it registered whenever it wants. The second level of access is *single authentication*. When the mashup application requests user data, the mashup application needs to redirect the user to the data provider, and the data provider will ask the user to authenticate him or her. After that, the data provider will ask the user whether he or she authorizes the mashup application to access all the user data that the mashup application has in its registration file. The user can choose the time period of authorization: for example, 1 hour, 3 hour, or 24 hours. Within that time limitation, the mashup application can access any user data in its registration file. The third level is *individual authentication*. The user can specify which user data require an individual authentication, so when a mashup application tries to access this data, it will always require user authorization.

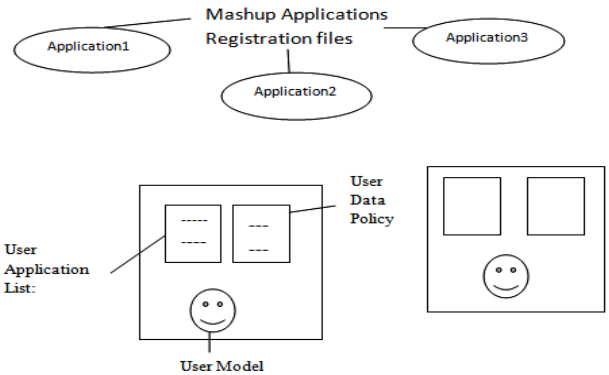

Fig. 4. The Components of the mechanism

4.3. User applications list

Each data provider hosts a list of mashup applications which have requested to receive user data for each user about whom the data provider keeps a user model. In that list, the user can see overview of the all mashup applications that the user has authorized. The user can discontinue or change the authorization at any time.

4.4 User Data Policy

To facilitate user decision making, the user can save his or her privacy settings. The User Data Policy is a file hosted at the data provider system. It has two parts: *policy about application providers* and *policy about data usage*. The *policy about application providers* contains a list of trusted application providers and a list of blocked application providers. Note that this list contains providers, not individual applications. Users can view and manipulate these lists based on external information, e.g. provider reputation services, press, etc, which may change their level of trust in particular application providers. Some providers, of course, may be unknown to the user, and not be in either of the two lists. In the *policy about data usage*, users can classify the data kept about them by the data provider into three levels: open-level, important-level, and crucial–level. The open-level data is accessible to all application providers except those in the blocked provider list. The important-level data is only open to the application providers on the "trusted" list. The crucial-level data is not to be undisclosed to any providers. The user can change both parts of the User Data Policy at any time. The purpose of the User Data Policy is to facilitate authorizing new mashup applications. When a user is authorizing a new mashup application, the data provider system can automatically compare the mashup application registration information with the User Data Policy to see whether there is a conflict. For example, the mashup application requires important-level data, but the application provider is not on the user's trusted list. If there is no conflict, the application will be authorized. Otherwise, the user data provider system will inform the user about the conflict, and the user can decide whether to change the policy (add the application provider to the trusted provider list) or not authorize the application. If the mashup application requires crucial-level data, the user has the choice of rejecting the authorization or still allowing it by changing the user data policy by moving the data to important level and adding the application provider to the trusted list.

4.5 Update application

Mashups can change their requirements for user data at any time. An updated mashup application has to update its registration information at all user data providers from which it receives user data. Data providers maintain version control on mashup application registration and user application list. The registration file of a mashup application keeps a version number for the application. In the user's application list, the mashup applications are also listed with their version numbers. When a mashup application updates its registration information, the data provider will increase the mashup application's version number. So, the version numbers for this mashup in its registration and in a user's application list will not match anymore.

4.6 Overall Workflow

Every time, when mashup application tries to access user data from the data provider, the data provider will check if the versions of the registration and user's application list match. If they do not match, the data provider will request user authorization. The mashup application will redirect user to the data provider, the data provider will show the update of the mashup registration to the user, and ask for authorization. If the user authorizes the updated mashup application, the version number on the user's application list will be updated according to the registration information. When a mashup application requests user data, first the data provider will check the mashup application's API and secret key. After that, the data provider will check whether the user data mashup application has requested is listed in the mashup application's registration. After that, the data provider will check whether the mashup application is on the user's application list. After that, the data provider will check if there is a match between the versions of the mashup application registration information and the user's application list. In the final step, the data provider will check the authorized access rights of the mashup from the user's application list.

5 Discussion

The proposed mechanism is user-centric; the user can see what kind of data is required for the mashup application and can authorize the mashup application's access to user data in a flexible way. The proposed mechanism refines the authentication process. The user can control his/her level of involvement in the authentication. If the user trusts the mashup application, he or she does not need to be involved in authentication at all. If user wants, he or she can control each step of user model interoperation. The user can chose to control only on the sensitive data's interoperation. Comparing with Shibboleth, this mechanism is light-weight; it does not require installing any software. It is ideal for small and middle-level application.

This mechanism also has some limitations. It makes the authorization process more complex. It requires more user involvement the first time when the user uses a mashup application. It also puts limitations on the mashup application development. Developers have to openly declare what kind of user data is required. Developers of applications that share user data and serve as data providers have to implement the components of the mechanism (see Fig. 4): a component that receives and updates the registration files of mashup applications, the user application list and user data policy, as well as an interface for the user to view and modify the user application list and user data policy.

The impact of mashup performance is not clear yet. If the user grants the mashup application the highest access rights, the performance should be the same as without the mechanism. But if user requires individual data authentication, the performance would be worse. Yet the user may be willing to accept the worse performance in exchange for enhanced privacy. The scope of the mechanism does not allow it to enforce how the mashup application treats user data. In the registration, the mashup application has to declare how it is going to treat the data: how long it will keep it, whether it will transfer the data to other parties or not, if it will disclose the data to other users or not. However, this mechanism cannot enforce the mashup

application's compliance to its own declaration. Trust and reputation management mechanisms can be used as an orthogonal approach for ensuring that mashup application providers have an incentive to treat user data according to the registration.

Finally, social network systems and existing mashup applications on the social web store also a lot of user-contributed data that can be used later in data-mining to develop new user profile data, not explicitly represented at the moment of sharing. It is an open question how to handle the potential privacy threats arising from harvesting user-contributed data.

The data about the user's social network presents further issues. So far we were talking about exchanging user data across applications only, but these applications typically have many users. Will these users be allowed to see the user's data or not? Can users define rights for accessing data to their friends / social network that can be propagated from one application to another? Some work on sharing data in blogs addresses this issue [14].

6 Future Work

We have implemented the proposed mechanism in a mock-up social network site environment. We plan to design several scenarios involving some sensitive user data and do an evaluation of the mechanism with real users based on these scenarios. There are several hypotheses we want to test during the evaluation: First, that the user data that can be shared is shown to the user in an understandable way. Second, that the user can easily express his or her privacy control settings through the User Data Policy. Third, that the user understands from the mashup application registration file (displayed in an appropriate way) why the application needs his or her model, the benefits for user model interoperation and how the application treats the user data. Fourth, this mechanism should help to increase user participation with respect to adding new mashup applications in an experimental group that uses the framework, in comparison with a control group which use the traditional authentication and privacy mechanism. We hope to be able to test these hypotheses with a large number of users on a social network site and will use questionnaires and collect statistics about user's participation that will be analyzed to validate or refute the hypotheses. In the next stages, we will combine this mechanism with services for user model data semantic translation, service discovery, and user identity mapping mechanisms to achieve a complete user model interoperation framework.

7 Summary

Personalized mashups provide a new way to do user model interoperation. Current mashup solutions face several challenges, including insufficient authentication and privacy control. This paper proposes a user-centric mechanism to facilitate authentication and improve user privacy control. Sharing user data on the social web raises many important issues. This mechanism addresses the privacy of sharing user model data that is explicitly represented by the application.

References

1. Thang, M.D., Dimitrova, V., Djemame, K.: Personalised Mashups: Opportunities and Challenges for User Modelling. In Proceedings of User Modeling. 2007, 415-419. (2007)
2. Musser, J.: Open APIs: State of the Market presentation. Qcon San Franciso (2008) http://qconsf.com/sf2008/file?path=/qcon-sanfran 2008/slides//JohnMusser_Web_As_Platform.pdf
3. Liu, H., Mae, P.: InterestMap: Harvesting Social Network Profile for Recommendations. Proc. 10th International Conference on Intelligent User Interfaces (2005)
4. Carmagnola, F., Dimitrova, V.: An Evidence-Based Approach to handle Semantic Heterogenity in Interoperable Distributed User Model. Proc. 13th International Conference on Intelligent User Interfaces (2008)
5. Recordon, D.: "Blowing up" Social Networks by Going Open (2008) http://www.slideshare.net/daveman692/blowing-up-social-networks-by-going-open-presentation
6. A.P. Sheth, K, Gomadam, J. Lathem.: SA-REST: semantically interoperable and Easier-to-Use services and Mashup. IEEE Internet Computing, vol.11, no.6, 2007, pp.91-94 (2007)
7. Reed, D., Chasen, L., Tan, W.: OpenID identity discovery with XRI and XRDS ACM International Conference Proceeding Series, Vol.283 Proceeding of the 7th symposium on identity and trust on the Internet Pages 19-25 2008, ACM, New York,USA (2008)
8. Berkovsky, S., Kuflik, T., Ricci, F.: Mediation of user models for enhanced personalization in recommender systems. In Proceedings of User Model. User-Adapt. Interact. 245-286. (2008)
9. Heckmann, D., Schwartz, T., Brandherm, B., Kroner, A.: Decentralized User Modeling with UserML and GUMO. *Proceedings 10th International Conference on User Modeling*, (2005) LNAI 3538 Springer, Berlin-Heidelberg, pp.428-432
10. Kobsa, A.: Privacy-Enhanced Personalization. Communication of ACM, vol.55, pp24-33 ACM, New York, NY, USA (2007)
11. Anwar, M.,Greer,J.: Role and Relationship-Based Identity Management for Private yet Accoutable E-Learning, Trust Management II, vol.262, pp.343-358 Springer ,Boston (2008)
12. Cranor,L.F., Reagle,J., Ackerman, M.S.: Beyond Concern: Understanding net users' attitudes about online privacy. AT&T labs-research technical report TR 99.4.3, http://www.research.att.com/library/trs/trs/99/99.4/. (1999)
13. Macquarie E-Learning Center of Excellence (MELCOE), Macquarie University: Shibboleth , October-November 2008 , AAF Shibboleth Rollout Workshop. http://www.federation.org.au/workshop/AAF%20Shibboleth%20Rollout%20Workshop%202008.1.pdf
14. Kay,J.: Lifelong Learner Modeling for Lifelong Personalized Pervasive Learning. IEEE Transactions on Learning Technologies Volume 1, Issue: 4 (2008)
15. Indratmo., Vassileva,J.:A Usability Study of an Access Control System for Group Blogs. Proc. ACM International Conference on Weblogs and Social Media. Boulder, CO (2007)