An OpenSocial Extension for Enabling User-controlled Persona in Online Social Networks

(Invited Paper)

Rodrigo Lopes, Hakan Akkan, William Claycomb, and Dongwan Shin
Secure Computing Laboratory
Department of Computer Science and Engineering
New Mexico Tech
Socorro, NM 87801
Email: {rodrigo, hakkan, billc, doshin}@nmt.edu

***Abstract*—User privacy is a challenging issue that must be addressed urgently in current online social networking (SN) sites. One of the fundamental problems associated with the issue is the lack of support of a user-centric approach to managing and sharing user profile information in current SN systems. In this paper we present a user-centric approach based on a credential system to enabling a user-controlled attribute (persona) sharing in online SN sites. Specifically we extend a Google-initiated open source project called *OpenSocial*, which provides a framework to support user attribute sharing between gadgets and online SN sites, in order to allow users to selectively share their attributes among online SN sites. This paper details the design and implementation of our extension.**

***Index Terms*—user privacy; social network; opensocial;**

I. Introduction

Online social networking (SN) sites have rapidly emerged, diversified, and adopted in recent years, and the wide adoption of the Internet contributed to the recent thriving popularity of those SN sites. The fundamental building block for the proper operation of most of existing social networks is personal information or attributes. To support this, most of online SN sites collect and process information regarding their entities, generally individuals, and offer a variety of features such as personalization, affinity sharing, and novel services enabled by the third party gadgets [1]. For instance, MySpace and Facebook allows millions of individuals to create personal profiles and share personal information with vast networks of friends. Hence, SN sites can create a central repository of personal information, which is persistent and cumulative [2]. Consequently, marketers, school officials, government agencies, and online predators can collect data about users through online SN sites. It is strongly believed that one of the most challenging problems in existing SN sites is related to this issue, *privacy*, and it must be addressed urgently.

One of the fundamental problems associated with the issue of privacy is the lack of support of a user-centric approach to managing and sharing user profile information in current SN systems. The basic notion of user-centricity is to give users, not organizations, a larger degree of control over personal information, and it has been practiced widely in the federated identity management (FIM) domain to provide a better mechanism for upholding user privacy over identity attributes [3], [4]. In order to support the notion of user-centricity in existing online SN systems, we identified three fundamental services, which are identity attribute management, privacy preference management, and selective attribute sharing [5]. Further we proposed a framework called U-Control that enables those services, as shown in Figure 1. This paper mainly concerns the selective attribute sharing component in the framework.

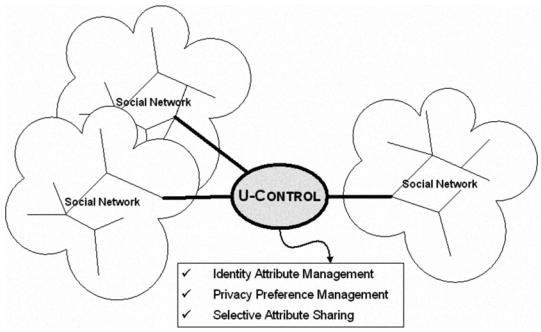

Fig. 1. U-Control Framework

In this paper we present a user-centric approach based on a credential system to enabling a user-controlled attribute (persona) sharing in online SN sites. Specifically we extend a Google-initiated open source project called *OpenSocial*, which provides a framework to support user attribute sharing between gadgets and online SN sites, in order to allow users to selectively share their attributes among online SN sites.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 describe the selective attribute sharing approach through the use of a credential system, followed by the discussion of our design and implementation in Section 4 & 5, respectively. Section 6 concludes this paper.

II. OpenSocial and Related Work

The OpenSocial project [6] was started to provide application developers with a single API that would allow the development of applications that would run across different websites, addressing the issue of a different social application API for each SN site. The OpenSocial API, implemented by

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8401
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8401

each provider, would allow a developer to build an application once and deploy it on different websites without any modification to the code. On the other hand, it would also allow upcoming SN sites to have support to a diverse portfolio of applications already developed for popular and established SN websites, and at the same time saving the time and effort of designing and building their own social application API.

The API is divided in three main areas according to the functionality provided. These areas are People and Relationships, Activities and Persistence. People and relationships provide functionality to access users' information and the relations that users have with each other. The activities functionality allows the application developer to access the users' activities in the context of the website and finally the persistence API provides functionality for the application to store and retrieve application specific information.

The API views users from three different perspectives: viewer, owner and friends. The viewer is the person in whose browser the application is being rendered and displayed while the owner is the person who owns the profile or information being displayed. Friends refer to the social connections of either the viewer or the owner. The owner and viewer may be the same person if, for example, someone is looking into his own profile and an application is running on that profile, using the owners information who happens to be also the one person looking at the displayed web page.

One of the main limitations in the OpenSocial is the inability for an application to access user information from different websites. Recently, different online SN sites have proposed mechanisms to allow the user profile to be exported to other websites, social networking or not. Some of these initiatives are MySpace Data Availability and Facebook Connect [7], [8]. However, the sharing of information on these initiatives is one way, in other words, the information can be shared from the SN website with other websites, but not vice-versa. Also, the source SN website is directly involved with the sharing of user profiles, gaining the knowledge of all the services the user is sharing his information with, and thus having an implication of privacy violation.

A. Credential Systems

A credential system is a system in which a user can obtain credentials (i.e., signed statements) from one organization and demonstrate possession of them to other organizations, and several credential systems based on number-theoretic schemes have been proposed for different purposes in literature. Chaum's approach to designing the digital cash system [9], [10], based on blind signature techniques, was one of them, also called an anonymous credential system. This system made it possible to obtain a certified statement from an issuer and show it to a verifier without the possibility of tracing the use of credentials. The credential was built upon the number theory, especially the use of an exponent that defines the type and the value of the credential in the blind signature. One major disadvantage of using this system is that a trusted third party is always required that all participating entities are dependent upon.

Similar to Chaum's system, but a more advanced scheme to design an anonymous credential system was presented by Brands [11]. His credential system could support many features such as expressions of any satisfiable proposition from proposition logic, limitation on the number of times a credential may be used, revocable anonymity, and discouragement of lending credentials. Camenisch et al. [12], [13] proposed a credential system that relies heavily on proofs of knowledge like Brands' system. One of the main disadvantages in these credential systems is obviously related to the expensive computational aspect of their cryptographic primitives using number theory and zero-knowledge proof (ZKP).

III. Selective Attribute Sharing using Authenticated Dictionary

We proposed an authenticated dictionary (ADT)-based approach to designing a credential system that allows users to selectively disclose their attributes [14], [15], and our approach is based on skip lists. A skip list is a data structure that allows the effective search and update of elements within a set. For more about the skip list, please refer to [16].

Our proposed ADT is used to represent a credential holding user attributes. Further, the credential allows the user to disclose a subset of his attributes to a verifier. Specifically, personal attributes and corresponding random values are hashed, ordered, and stored in the skip list as elements. Hence, ADT will not contain any user information in it. Additionally, to prevent an offline dictionary attack on the hash value based on the limited domain values of some attributes, personal attributes are salted. The running time complexity of ADT is $\mathcal{O}(\log n)$ for both verification and update, thereby making its implementation very efficient. Issuing and showing credentials are as follows. After establishing a pseudonym, the user requests the issuer to issue a credential containing the attributes the issuer can assert about the user. The issuer will calculate the hash value of each of the attributes and a corresponding random value that will be included in the credential and then order them by the hash values. From here, ADT can be built, with the last node signed by the issuer's private key. The issuer then sends the credential to the user along with attribute and random value pairs. To show a credential to a verifier, upon the request for personal attributes required for a service, the user sends the set of pairs of requested attributes and random values, and corresponding hash values for each of the attributes to allow the verifier to verify the attribute. This set of attributes corresponds to the actual value of the attributes. The verifier will hash the attributes and corresponding random values to verify that they are in fact part of ADT.

IV. OpenSocial Extension: Design

Our extension of OpenSocial mainly concerns the support of an ADT-based credential system to allow users selectively share their attributes among OpenSocial-based SN sites as well as gadgets running on top of them. Though our initial

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8401
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8401

design for an OpenSocial extension was proposed in [14], it was limited in a sense that the design of a credential issuing component was not addressed and the design of a data structure was inefficient. The current design of our extension approach addresses both of these issues to seamlessly support the issuance and usage of an ADT-based credential in OpenSocial-based SN systems.

A. API Extension

We extended the functionality that allows user attributes to be retrieved from an OpenSocial container in order to accommodate the ADT-based credential functionality with the minimal possible changes to the exposed interfaces in OpenSocial. The data type provided by the OpenSocial specification to allow gadget developers to obtain user attributes is the DataRequest object, which also contains all the necessary functions to set up request details, perform the request, and specify the function that should be called upon response reception. We directly extended this available object, overriding the relevant functions, which are all except the ones relative to application data, which will be addressed in the subsection B. As shown in Figure 2, our extension class, DataRequestEx, subclasses DataRequest, inheriting all the functionality available in the original type and changing solely the relevant function that calls the appropriate entity to handle the request made by the social gadget. Note that our extended object is slightly different from ones proposed in [14].

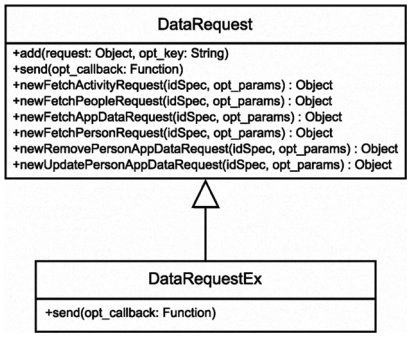

Fig. 2. Revised UML representation of DataRequest and DataRequestEx Objects and their relationship

In addition, our extension of OpenSocial should address the ability to issue credentials. To this end we want to make available to the container website a set of functionality that allow the service to add issuing capabilities, by simple calling the functionality provided by OpenSocial. This functionality allows the container to offer the user the possibility of requesting a credential, and, according to the container definitions, issue a credential containing all OpenSocial compatible information for the user available in the service or allow the user to specify what information to be included in the credential.

B. Data Structure Extension

As it is one of the main goals of OpenSocial to provide a framework enabling developers to create social gadgets that

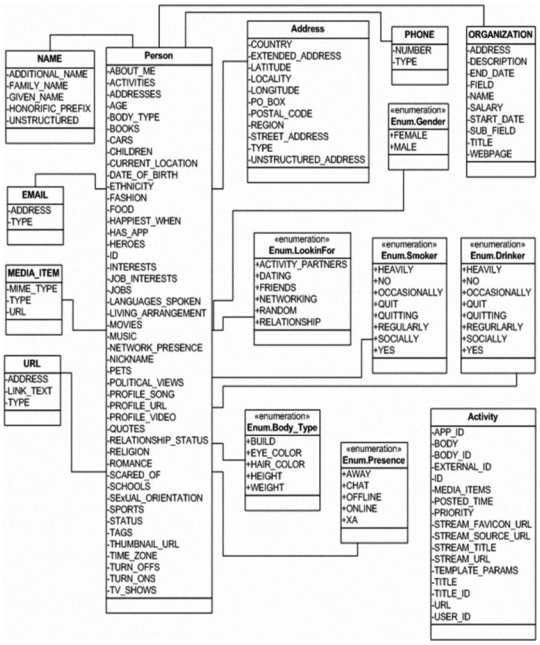

Fig. 3. OpenSocial Specification Person and Activity UML Diagrams

can run on any service implementing the specification, the data definition has been specified to be as broad as possible, delegating to services the mapping between their own data structures and the OpenSocial Data Model (ODM). The ODM includes information about the users, the relationships between them and the activities. The ODM also includes another set of information related to applications, which would allow applications to store application specific data in the container service. This application specific data is defined very vaguely in the specification and is inherently dependent on the container implementation decisions, namely if it is to be supported and what amount of data should be made available to an a social gadget. Since this data type is not relevant in the context of our proposed extension, we will not cover it any further. Other types that are very relevant to our extension are Person and Activity. The Person type includes the basic information about an individual, in the form of different attributes. The Activity type contains information about particular activities or events that were created or happened in the container service. The connections between the different users are represented by relationships between different instances of Person. Figure 3 shows an UML diagram representing the types of People and Activity, as derived from the OpenSocial Specification.

In an ADT-based credential to be supported in our extension of OpenSocial, attributes are represented as type-value pairs: ⟨type:value⟩. In order to accommodate the ODM types, it is necessary to make a mapping between these types and the attributes inside the credential. Our design goal is to make the control granularity the user has over the information being shared as fine as possible so that the user can share only some fields of attributes that he wants to be shared. To achieve this goal, we define the ODM objects to be a hierarchy of attributes

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8401
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8401

inside the credential. A single type instance will not be a single attribute in the credential, but rather a group of attributes. The ODM types that can be the top level types of a hierarchy are only, in our context, Person and Activity. The relationship level we consider is only zero or one, that is, a credential will only contain information on the owner of the credential or friends of the owner of the credential, that is, people that have direct connections with the user for whose profile the credential was issued. To this end an extra attribute is added to our credential which indicates which person instance in the credential represents the person who owns the credential. All other person instances included in the credential will represent people with whom the owner person has direct connections within the issuer service context.

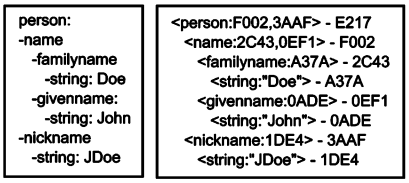

Fig. 4. Sample Person instance and its mapping to an ADT credential

Detailed in Figure 4 is how a sample person object would be deconstructed into its basic types and how the hierarchy would be achieved. In this example we have the person instance for an individual named John Doe. This person instance is composed of the attributes of name and nickname, and the name itself is a type composed of two strings, givenname and familyname. The sample information contained on the person instance is represented on the left side of Figure 4. On the right side of the figure is the representation of all the attributes that would be contained by a credential to represent the person instance. As it can be deducted, the instance is inserted in the credential bottom up, with the most basic types being inserted first and the more complex types being inserted later, and with values depending on the more basic attributes previously added to the credential. It can also be observed that the complex types have a value that does not correspond to the actual value of the type but to a list of hashes of the more basic types that comprise them. The value found for the person attribute is in fact a list of two hash values, one for the name type and the other for the nickname type. This representation allows a very fine granularity. For example, it would be possible to disclose and share only the familyname without disclosing anything else because it can be easily seen that the familyname attribute is a composing part of name that is a composing part of person, and this would be done without the need to show any other information. It would not be possible to add information to the person object, because that would imply that we could alter the value of the attribute person and add a new hash value for the information an attacker may try to inject, of course inserting a new attribute into an existing credential could not be possible, but using the value of an existing attribute as a part of other attribute it does not actually belong to could be attempted. This will not work, because each type has references to all the attributes that compose it or, the value itself in the case of primitive types.

V. OpenSocial Extension: Implementation

Our implementation is based on Shindig, which is a Java-based implementation of OpenSocial, originally started by Google and now hosted and managed by the Apache Software Foundation. The Shindig project was developed upon an existing code base from Google and reuses many components already in use for gadget rendering in their popular portal iGoogle. Although a PHP version of Shindig is available, we implemented our extension of OpenSocial in Java, since our prototype implementation of the credential system was also developed in Java.

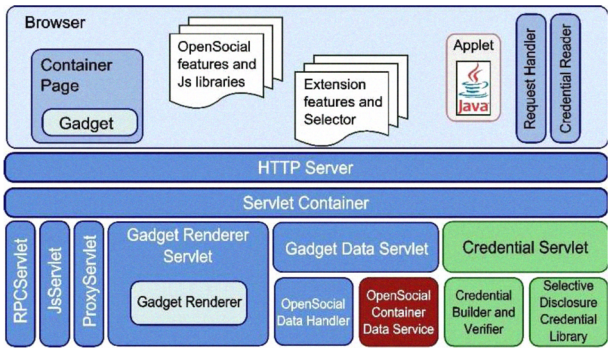

Fig. 5. Server-side Extensions within Shingid

The main OpenSocial related component in Shindig is the Gadget Data Servlet. This component handles and replies to the request originated inside the social gadgets running on the container SN site. The OpenSocial specification, as the functionality exposed to developers of social applications, is implemented as JavaScript libraries that are included in the gadget when it is rendered and executed on the container.

A. Server-side Implementation

We implemented the functionality to issue credentials at the core of the Shindig implementation, side by side with the Gadget Data Servlet. All the services required to make the credential issuing service work at the server-side is provided by the Credential Servlet, which is added to the Shindig architecture, as shown in Figure 5. This servlet exposes the functionality to issue credentials as well as the functionality that allows signature verification on credentials being used by the user to provide data to gadgets' requests. The credential signature verification is one of the most important steps in the credential usage, as it will verify the authenticity of the credential and the acceptance of a relationship of trust between the container service and the issuer service. The two main components used by the Credential Servlet, also shown in Figure 5, are the Credential Builder and Verifier and the libraries implementing the ADT based credential, which are used to generate the credential from the data obtained at the container.

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8401
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8401

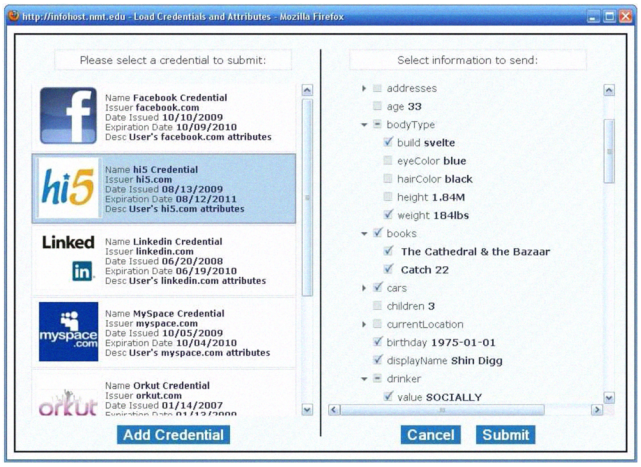

Fig. 6. Client-side Credential Selector

B. Client-side Implementation

The extension functionality on the client side, the code called by the gadgets and by the container to issue credentials, was implemented as JavaScript libraries, and it was added to the set of features already implemented in Shindig. The implemented functionality, to use a credential as a data source, ultimately sends the requests built by gadgets to a Java applet. This applet contains the logic to manage the user's credentials, handle the request, process the user's options, verify the credential authenticity by calling the credential verifier on the server, build the response object and send it back to the JavaScript function that will finally return the response containing the data, to the gadget. Communicating directly with the applet and being a liaison between user and the credential logic is the credential selector, which is shown in Figure 6.

One of the key components in our implementation is the credential selector, which is the only component the end user will have any contact with, apart from the credentials themselves. The credential selector allows the user to make choices on the information being released. This credential selector is included in the JavaScript libraries and is embedded in all gadgets that require our extension as a feature to be use. The credential selector will allow the user to add new credentials by choosing credential files from disk, and then displaying all added credentials so that the user can choose a credential for any particular request.

VI. Conclusion and Future Work

This paper discussed an approach to enabling a user-controlled attribute sharing in online SN sites. Our approach was based on an authenticated dictionary (ADT)-based credential system and extended an existing open source-based social networking framework called *OpenSocial* to support the credential system. We discussed the design of our extension attempt. Finally we presented a proof-of-concept implementation based on our design.

Revocation is an important issue in any credential-based system. Currently revocation is supported in our system in two ways: 1) using validity periods and 2) using online revocation systems like in PKI systems. We will investigate how to remove this online dependency. Last not but least, unlinkability is currently not fully supported in our current prototype system, and we will investigate how this can be supported.

Acknowledgment

This work was supported at the Secure Computing Laboratory at New Mexico Tech by the grant from the National Science Foundation (NSF-IIS-0916875).

References

[1] R. Gross, A. Acquisti, and H. J. H. III, "Information revelation and privacy in online social networks," in *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, Alexandria, VA, November 7 2005.

[2] F. B. Vigas, "Blogger's expectations of privacy and accountability: An initial survey," *Journal of Computer-Mediated Communication*, vol. 10, no. 3, 2005.

[3] G.-J. Ahn and J. Lam, "Managing privacy preferences for federated identity management," in *Proceedings of the 2005 Workshop on Digital Identity Management*, Alexandria, VA, USA, November 11 2005, pp. 28–36.

[4] D. Shin, G.-J. Ahn, and P. Shenoy, "Ensuring information assurance in federated identity management," in *Proceedings of the 23rd IEEE International Performance Computing and Communications Conference*, Phoenix, Arizona, April 14-17 2004.

[5] D. Shin, R. Lopes, W. Claycomb, and G.-J. Ahn, "A framework for enabling user-controlled persona in online social networks," in *Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference*, Seattle, Washington, July 20-24 2009.

[6] *OpenSocial Foundation*, http://www.opensocial.org.

[7] *MySpace Data Availability (DA)*, http://developer.myspace.com.

[8] *Facebook Connect*, http://developers.facebook.com/connect.php.

[9] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Communications of the ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[10] ——, "Achieving electronic privacy," *Scientific American*, pp. 96–101, August 1992.

[11] S. Brands, *Rethinking Public Key Infrastructure and Digital Certificates - Building in Privacy*. MIT Press, 2000.

[12] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *Proceedings of 3rd Conference on Security in Communication Networks*, Amalfi, Italy, September 12-13 2002.

[13] J. Camenisch and E. V. Herreweghen, "Design and implementation of the idemix anonymous credential system," in *Proceedings of 9th ACM Conference on Computer and Communication Security*, Alexandria, VA, November 7-11 2002.

[14] D. Shin and R. Lopes, "Enabling interoperable and selective data sharing among social networks sites," in *Proceedings of the 3rd International Workshop on Trusted Collaboration*, Orlando, Florida, November 13 2008.

[15] D. Shin, R. Lopes, and W. Claycomb, "Authenticated dictionary-based attribute sharing in federated identity management," in *Proceedings of the 6th International Conferecne on Information Technology: New Generation*, Las Vegas, Nevada, April 27-29 2009.

[16] A. Anagnostopoulos, M. T. Goodrich, and R. Tamassia, "Persistent authenticated dictionaries and their applications," in *Proceedings of 4th International Conference on Information Security*, Malaga, Spain, October 1-3 2001.

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8401
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8401