Available online at www.sciencedirect.com

SciVerse ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Access control for online social networks third party applications

Mohamed Shehab [a,*], Anna Squicciarini [b], Gail-Joon Ahn [c], Irini Kokkinou [d]

[a] University of North Carolina at Charlotte, United States
[b] Pennsylvania State University, United States
[c] Arizona State University, United States
[d] Indiana University-Purdue University Indianapolis, United States

ARTICLE INFO

Article history:
Received 31 May 2011
Received in revised form
28 June 2012
Accepted 11 July 2012

Keywords:
Access control
Social networks
Applications
Attribute generalization
Finite state machine

ABSTRACT

With the development of Web 2.0 technologies, online social networks are able to provide open platforms to enable the seamless sharing of profile data to enable public developers to interface and extend the social network services as applications. At the same time, these open interfaces pose serious privacy concerns as third party applications are usually given access to the user profiles. Current related research has focused on mainly user-to-user interactions in social networks, and seems to ignore the third party applications. In this paper, we present an access control framework to manage third party applications. Our framework is based on enabling the user to specify the data attributes to be shared with the application and at the same time be able to specify the degree of specificity of the shared attributes. We model applications as finite state machines, and use the required user profile attributes as conditions governing the application execution. We formulate the minimal attribute generalization problem and we propose a solution that maps the problem to the shortest path problem to find the minimum set of attribute generalization required to access the application services. We assess the feasibility of our approach by developing a proof-of-concept implementation and by conducting user studies on a widely-used social network platform.

1. Introduction

The recent growth of social network sites such as Facebook, Twitter and MySpace has created many interesting and challenging security and privacy problems. In social networks, users manage their profile, interact with other users, and self-organize into different communities. Users profiles usually include information such as the user's name, birthdate, address, contact information, emails, education, interests, photos, music, videos, blogs and many other attributes. Controlling access to the information posted on user profile is a challenging task as it requires average Internet users to act as system administrators to specify and configure access control policies for their profiles. To control interactions between users, the user's world is divided into a trusted and a non-trusted set of users, typically referred to as *friends* and *strangers* respectively. Furthermore, some social networks allow users to further partition the set of friends by geographical location, social group, organization, or by how well they know them. Users are provided with group based

* *Corresponding author.*
E-mail addresses: mshehab@uncc.edu (M. Shehab), acs20@psu.edu (A. Squicciarini), Gail-Joon.Ahn@asu.edu (G.-J. Ahn), ikokkino@iupui.edu (I. Kokkinou).
0167-4048/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cose.2012.07.008

access control mechanisms (Facebook Inc, 2011) that apply access rules on the different groups of friends and strangers. Facebook, one of the most popular social sites, enables users to create friend lists and to compose profile policies based on these friend lists (Facebook Inc, 2010). In addition to the challenges involved with enabling fine grain access control for user profiles (Damiani et al., 2002) to control which data attributes viewable by other users, a yet unexplored problem is related to users' profile access from entities different from other social network users.

With the development of Web 2.0 technologies (O'Reilly, 2005), online social networks are able to provide open platforms to enable the seamless sharing of profile data to enable public developers to interface and extend the social network services as applications (or APIs). For example, Facebook allows anyone to create software plug-ins that can be added to user profiles to provide services based on profile data. Although these open platforms enable such advanced features, they also pose serious privacy risks (Tootoonchian et al., 2008; Gates, 2007; Hart et al., 2007). Users' profiles in fact have a great commercial value to marketing companies, competing networking sites, and identity thieves.

Social networks platforms have focused on user-to-user fine grain access control, for example, the Facebook Privacy Policy allows users to specify fine grain policies controlling which profile attributes can be accessed by their friends and friends of friends (Facebook Inc, 2009). When installing social network applications users have to grant the applications all the requested permissions in order to successfully complete the installation process (Facebook Inc, 2010; Myspace Inc, 2009). For example, Fig. 1(a) and (b), show the application permission request displayed by the Google+ and Facebook platforms respectively when the user attempts to install an application. Basically, the adopted application access control model is an *all-or-nothing* policy, where the application should be granted all the requested permissions in order to install it successfully. In addition, API developers have access to users' data regardless of the actual applications' needs, leading to potentially serious privacy breaches (Irvine, 2008; CNET Blog, 2008; Washington Chronicle, 2008). Such privacy threat is often hidden or not clear to social network users, who are often not aware of the amount of data that is actually being disclosed, since they do not really distinguish between social network users and developers outside the social network boundaries. In November 2011, Facebook's privacy practices were the subject of complaints filed with the Federal Trade Commission (FTC) (2011). The complaints were related to the Facebook's privacy practices that deceived customers and failed to keep privacy promises. One of the main complaints was related to Facebook's claim that third-party applications that users' installed would have access only to user information that they needed to operate, where in fact, the apps could access nearly all of users' personal data. In addition, Facebook claimed that it certified the security of apps participating in its "Verified Apps" program, where in fact they did not.

We believe, in order to promote healthy development of social network environments and to protect individuals' privacy rights, users should be able to take advantage of the available applications while still having a stronger control on their data. The problem is not trivial, in that it requires designing new access control models for APIs in social networks, as well as extending social network applications. Applications should be designed and customized with the users' profile preferences, and users should have the ability to specify the data that they are willing to reveal. Additionally, users should be able to use data privacy mechanisms such as generalization to enjoy the services provided through APIs without having to disclose identifying or private information.

In this paper we address this issue by deploying an access control mechanism for applications in social networks. Our goal is to provide a privacy-enabled solution that is in line with social network ethics of openness, and does not hinder users' opportunities of adding useful and entertaining applications to their profiles. Our access control mechanism is based on enabling the user to specify the data attributes to be shared with the application and at the same time be able to

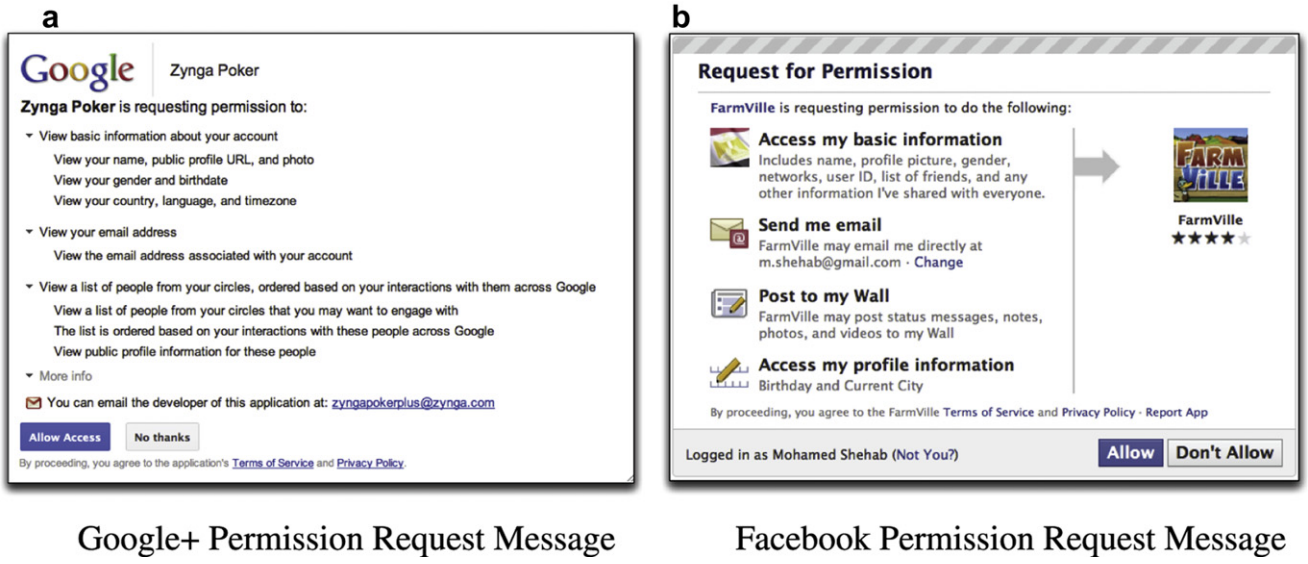

Fig. 1 – Social networks profiles and applications.

specify the degree of specificity of the shared attributes. Enabling such a mechanism requires applications to be developed to accommodate different user preferences. We model applications as finite state machines, and use the required user profile attributes as conditions governing the application execution. The user is faced with the challenge of specifying the minimum set of attributes and their minimum generalization levels required to acquire specific services provided by the application. In order to address this problem we proposed the weighted application transition system and formulated the Minimal Attribute Generalization Problem. Furthermore, we propose a solution that maps the problem to the shortest path problem to find the minimum set of attribute generalization required to access the application services. We assess our solution by implementing a proof-of-concept prototype using the Drupal platform, which is an open source platform for the development of online communities and social networks. Additionally, we conduct extensive user studies using the Facebook social network. We simulate our selective installation process for different applications currently provided by Facebook and assess the users' perceived benefits and ease of use. The response is encouraging and positive, in that respondents acknowledge the need for solutions of this kind to better protect their privacy and security. They also believed that our approach is appropriate to gain control of the data disclosed at the application's end.

The rest of the paper is organized as follows. In Section 2, we describe the related work. In Section 3, we provide background information related to Social Network APIs. In Section 4, we introduce our developer APIs access control framework. In Section 5, we discuss how to provide customized applications. Section 6 presents our implementation and experimental results. The conclusion and future work are discussed in Section 7.

2. Related work

Security and privacy in social networks is currently a well-studied research topic (IEEE. W2SP, 2008; Masoumzadeh and Joshi, 2011; Acquisti and Gross, 2006; Hogben, 2007; Golbeck and Hendler, 2006; Fong, 2011; Carminati et al., 2006, 2009). Several studies conducted in the past few years have identified the need for solutions to address the problem of information leakage in social networks. These solutions are envisioned to be based on interpersonal relationships and very flexible social interactions. In the following, we discuss some of the most relevant approaches developed along these dimensions.

Gollu et al. (2007) proposed a social networking access control scheme that considers user identities as key pairs, and social relationship on the basis of social attestations. They adopted access control lists to represent user access settings. A more sophisticated rule-based access control model has been proposed by Carminati et al. (2006). Such an approach is based on the enforcement of complex policies expressed as constraints on the type, depth, and trust level of existing relationships. The authors also proposed using certificates for ensuring the authenticity of user to user relationships, and client-side enforcement of access control according to their proposed rule-based approach, where a subject requesting access an object must provide the required authorization tokens to be granted access.

Along similar ideas, Hart et al. (2003), proposed Privacy-aware bLOGing engine (PLOG), that is automatic, expressive, and convenient. The authors focused on exploring content based access control for social networks to detect and deter malicious activities. Even though their main focus was on user to user interactions this work can be extended to accommodate social applications. The HomeViews (Geambasu et al., 2007) project is a related light-weight content sharing access control mechanism, which facilitates ad hoc, peer-to-peer data sharing between unmanaged home computers. Sharing and protection are accomplished without centralized management, global accounts, user authentication, or coordination of any kind. The approach is based on a peer-to-peer middleware system that simplifies the construction and management of distributed personal information sharing applications. With HomeViews, applications can easily create views, compose views, and seamlessly integrate local and remote views. Baden et al. have recently proposed Persona (Baden et al., 2009), which focuses on privacy issues in social networks by proposing a cryptographic-based approach. Persona hides user data with attribute based encryption, allowing users to apply fine-grained policies over who may view their data. The authors described an implementation of Persona that replicates Facebook applications and showed that Persona provides acceptable performance when browsing privacy-enhanced web pages, even on mobile devices. While notable, the proposed approach involves a level of sophistication from end-users which is unlikely, and it does not clearly address the issue of third-party applications.

Besmer et al. (2009), presented a new user-to-application policy which greatly restricts the information applications can access, while still allowing useful and desired information sharing. Their proposed model leverages user to user influence by presenting users with policy recommendations based on their friends' previous experiences when installing social applications. Singh et al. (2009) proposed the xBook framework which prevents untrusted third party applications from leaking users' private information. The xBook framework is based on controlling information flow between the different application components, based on a data labeling and application confinement. Felt and Evans (2008) proposed a novel solution for protecting privacy within social networking platforms through the use of an application programming interface to which independent application owners would agree to adhere to. This approach enables users to protect their information attributes, however the required agreement limits the wholesale adoption of a privacy proxy.

Other related work has analyzed both privacy risks associated with information disclosure in social networks, and developed initial mechanisms to protect against some involuntary information disclosure. Liu and Terzi (2009) proposed a framework for deriving a "privacy score" to inform the user of the potential risks to their privacy created by their activities and activities with other users within the social network. To mitigate against these information leakage channels, Vanetti et al. (2011), proposed a system for content-based message

filtering for social networks. The system allows users to automatically control the privacy of messages posted on their profile walls. The approach is based on a flexible rule-based system that allows users to customize their filtering criteria, and a supervised learning classifier that automatically manages the labeling of messages based on their content.

In regards to work specific to applications, Felt et al. recently reviewed the permissions requested by current applications (Porter Felt et al., 2011). While some of their findings apply to the context of Android applications, they confirm our claim that up-front permission requirements for installation may help applications achieve their full potential in a secure fashion, while still be useful for end-users. Finally, this work extends our preliminary research (Shehab et al., 2008), in which we first introduced the notion of access control for third-party applications. In this paper, we developed such ideas more in-depth, and validated in a two step fashion, by implementing it on an open source platform and by means of a user study, whereby we assessed the user's perceived value of our approach.

3. Background on social network APIs

With the emergence of new web technologies, and with the establishment of the Web 2.0, a large number of web sites are exposing their services by providing web programming interfaces (APIs). For example, Google Web API (Code, 2009) provides a programming interface to query web pages through Google from user developed applications. Several social network web sites have released APIs that allow developers to leverage and aggregate information stored in user profiles and provide extended social network services. The exposed APIs are basically a set of web services that provide a limited and controlled view for the application to interface with the social network site. The social network application architecture includes three interacting parties namely the user, social network server, and the third party application server. Fig. 2(a), shows the different blocks used in the social networks architecture. Note that the application server is able to connect to social network through the exported web APIs. Furthermore, these requests are filtered through the request management module which will be discussed in detail in the next section.

For example, consider an application that recommends stores in your area that are having sales. In this case, the application requires access to retrieve your address, age, marital status, and gender. The address information is required to be able to locate shops in your area, and the other parameters are required to provide a more focused recommendation. Some other applications would not only require data from your profile but would also require data from your friends' profiles. For example, consider an application that projects your friends on an online map according to the address listed on their profiles. This application requires your address and your friend list, then for each friend it would retrieve their address.

Social networks provide mechanisms for users to customize their profiles and to add applications developed by external developers. The application provides the customized services by accessing the exported APIs. Fig. 2(b), depicts the interaction stages between the user browser, social network and the third party application. The interaction starts when a user requests an application *APP* (Steps 1–2). The application server interacts with the social network server by instantiating API calls (Step 3). Upon receiving the responses of the API calls, the application server compiles and sends a response to the social network which is forwarded to the requesting user (Steps 4–5).

4. Developer APIs access control framework

Applications require permission to access user's profile data to provide a service customized to the user's profile data. In this section we present our approach to enable fine grain access control (Damiani et al., 2002; Rizvi et al., 2004) for third party applications, to limit applications' access only to relevant user's profile data. We first provide some preliminary definitions related to applications and API set, and then we discuss our proposed fine grain access control framework for API based applications.

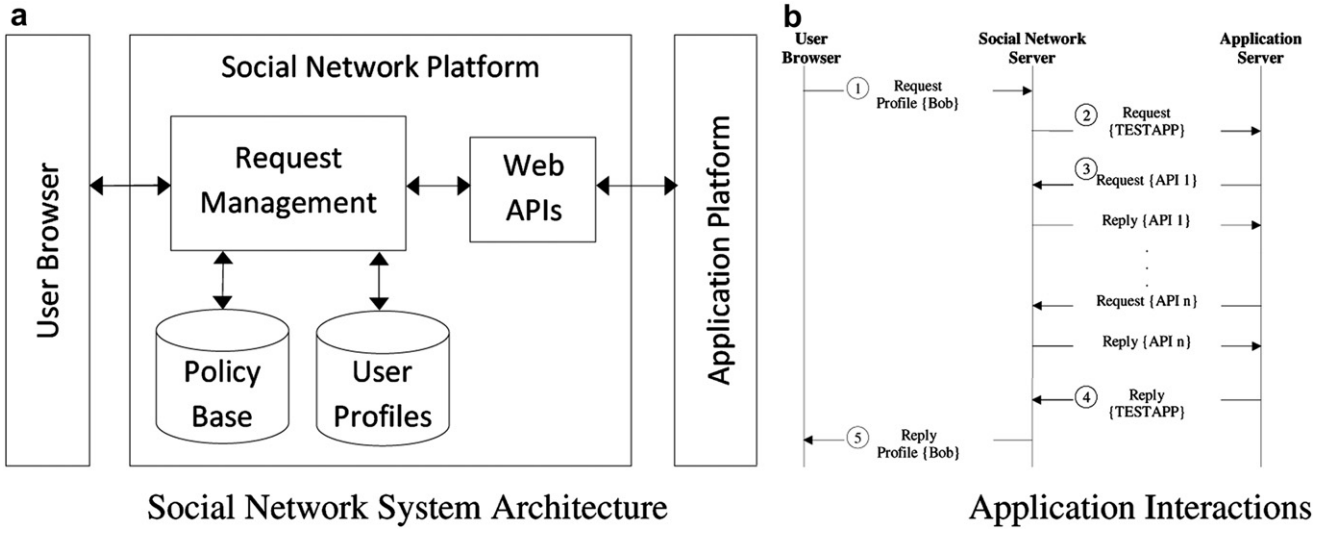

Fig. 2 – Social network architecture and application interactions.

4.1. Social network profiles and data sets

For the purpose of our work, the two main components of a social network are represented by users' profiles and applications.

Users' profiles. Users' profiles are modeled as collection of data items that are uniquely associated to them. Each data item is defined over a finite domain of values.

Definition 1. *(User Profile) A user profile for user i, is characterized by an attribute vector* $x = \{x_1, \ldots, x_n\}$, *where attribute* x_i *takes values in a domain* D^i, *which also includes the null value referred to by* $\perp$.

A common practice in privacy preservation mechanisms is to replace data records with suppressed or more general values (Samarati and Sweeney, 1998; Sweeney, 2002) in order to ensure anonymity and prevent disclosure of sensitive data. A simple disclosure policy can simply suppress an attribute if certain disclosure criterion are met, in this case that is a all or none policy. A generalization disclosure policy, is accomplished by assigning a disclosed value that is more general than the original attribute value. For example, the user can make the address information less specific by omitting the street and city and revealing just the zip code. Fig. 3, shows an example of a partial value generalization hierarchy of the address attribute. We assume that domain D^i for a certain data item x_i (see Definition 1) is a partially ordered set $(D^i_j, \prec)$, where D^i_j are the attribute generalizations and $\prec$ is the ordering operator. In the domain D^i the largest element corresponds to the non-generalized attribute value and the smallest element is the most generalized value which is the suppressed value $\perp$. The domain D^i contains l_i generalization levels, an attribute generalized to the h^{th} level of generalization is denoted by D^i_h, where $0 \leq h < l_i$. Data attribute generalized to D^i_1 is more general than an attribute generalized to D^i_2, $D^i_2 \prec D^i_1$, which implies that D^i_2 discloses more information than D^i_1.

Given a user profile x, by specifying generalization preferences for each of the profile attributes the user is able to specify a different view for each application. The user generalization preferences for an application is defined by the attribute generalization vector $UP = [h_1, \ldots, h_n]$ where h_i represents the generalization level D^i_h permitted for profile attribute i. Different attributes have different disclosure sensitivity, for example some users might regard their home address more sensitive than their cell phone number. To capture attribute sensitivity, for each profile attribute $x_i \in x$ the user assigns a sensitivity metric Φ_i, which is specified for the non-generalized attribute $D^i_{l_i-1}$. Note that the sensitivity of an attribute x_i generalized to level h_i is proportional to $\Phi_i h_i$. Given a user generalization preference vector the $UP = [h_1, \ldots, h_n]$, the risk of attribute disclosure is proportional to $\Theta(UP) = \sum_{i=1}^{n} \Phi_i h_i$. Note that the function $\Theta()$ provides a mechanism to compare user generalization preferences. In addition to the profile attributes the generalization model can also be applied to the tags and the metadata that are attached to the profile data.

Applications. *The building block for our model is represented by applications. Applications are composed of a set of API's which are functions called by the application.*

Definition 2. *(Application API Set). Given an application App, the application API set App.apiset is the set of APIs called by application App, represented as the set* $App.apiset = \{api_1, \ldots, api_n\}$.

For example, consider a horoscope application *HoroAPP*, illustrated in Fig. 4. It calls the "*user.get_birthday*() " and "*user.get_friends*() " APIs. The application API set for *HoroAPP* is $HoroAPP.apiset = \{user.get\_birthday(), \ user.get\_friends()\}$. From the API calls the set of data set accessed by the application can be obtained by tracing the data acquired by the called API's. For example, consider an API "*user.get_birthday*() ", the profile data accessed is {*profile.birthday*}. Other APIs involve the processing of several profile data items for example, consider the API "*user.get_photos_with_friends*() ", this API returns the photos taken with friends. The API performs a join between the user friends and the user photo album meta data, in this case the data items access are $\{profile.ablums, profile.friends\}$. Accordingly, an application can be translated from a set of API calls to a set of data accesses. This set of accessed data can be then presented to the user to select which data items to be exposed.

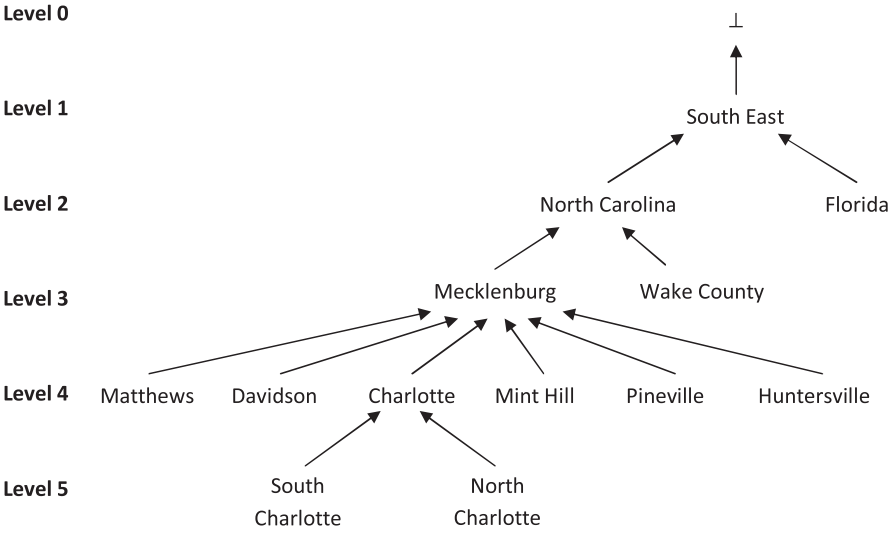

Fig. 3 – A partial value generalization hierarchy of the address field.

```
ExampleApplication(){
  a = get_friends(userid);
  ...
  b = get_albums(userid);
  ...
  Query = "SELECT birthday FROM user_db
           WHERE uid=userid";
  c = send_query(Query);
  ...
}
```

Fig. 4 – Example of horoscope application.

4.2. *The access control framework*

Our framework adopts the *Principle of Least Privilege* (Saltzer and Schroeder, 1975), which requires that each principal be accorded the minimum access privileges needed to accomplish its task. In our context, principals are the applications, which should be awarded access to the minimum set of profile data in order to provide the requested service. To achieve this goal we present a mechanism that enables fine grain access control on the profile data. Such a mechanism enables the application developer to select the data items required by the application and at the same time enables the user to opt-in or opt-out or generalize each of the requested data items. Specifically, our framework is characterized by three main phases: *application registration*, to register the application at the social network server; *user application addition*, to add the application in a local profile; and *application adaptation*, within which the application adapts according to the provided data items. We discuss them in what follows.

Application registration. The application developers register the application with the social network server. The developers are required to share the application API calls and the application business state diagram describing the application process, the details of this requirement will be discussed in following sections. As part of the registration process, developers need to tag the application, by labeling each API within the application with the set of user's data items used by the application. The tags provided during this stage only refer to the user's profile data involved and do not include any external output or additional user input that may be required when executing the API. The provided application information is used to compile an *application sheet* describing the data attributes required by the application.

User application addition. Once the application is registered with the social network server, it becomes available for social network users to add to their profiles. Upon selecting the application, the application sheet is presented to the user, who is prompted with the following options for each data item required by the API: choose to opt-in, opt-out, or generalize. Intuitively, the user opts-in for the data items he is willing to disclose to the application. If the user opts-out for some data the application needs to adapt in order to be properly executed without such input. In case the generalize option is chosen for a certain data item, then the user only accepts the application to employ generalized data attribute (Samarati and Sweeney, 1998; Sweeney, 2002). The user selections are input in the *user sheet*, which indicates the user access preference for the added application.

An example of XML encoding for the horoscope application sheet is reported in Fig. 5(a), where birthday, gender and address are requested. In Fig. 5(b) we report the user sheet in case the user opted to disclose only month and year of birth.

User application adaptation. At this stage the user sheet is used to generate a version of the application executable using the input obtained by the profile data items. This phase requires the application to differentiate provisioning according to the permissible data items and their respective generalization levels. We discuss in the next section how this not trivial task is achieved.

5. Customized application service provisioning

The user sheet provides a mechanism for users to specify generalization preferences on the profile attributes to restrict the data accessible to the application. On the other hand, by enabling attribute generalizations the application is faced with the problem of missing data, and might not ensure the provisioning of the request service based on the provided data generalizations. To address this issue we propose that during

a

```
<APPSHEET>
 <APP id="332198764">
  <DESCRIPTION>
   <NAME> Horoscope App </NAME>
   <INFO> Provide daily horoscope
    from www.horoscope.com </INFO>
  </DESCRIPTION>
  <DATA-GROUP>
   <DATA ref="birthday"/>
   <DATA ref="gender"/>
   <DATA ref="address"/>
  </DATA-GROUP>
 </APP>
</APPSHEET>
```

Application Sheet

b

```
<USERSHEET>
 <APP id="332198764">
  <ALLOW>
   <DATA-GROUP>
    <DATA ref="birthday.day"/>
    <DATA ref="birthday.month"/>
   </DATA-GROUP>
  </ALLOW>
 </APP>
</USERSHEET>
```

User Sheet

Fig. 5 – Application and user sheets.

the application registration phase the application developer is required to provide the process execution description of the application. The process execution description describes the interactions between the composed APIs. A candidate process description language standard is BPEL (Business Process Execution Language for Web Services, also WS-BPEL, BPEL4WS) (OASIS, 2008) which provides a rich vocabulary for expressing composition, orchestration and coordination of web services to describe the behavior of business processes. Fig. 6, shows an example process execution diagram describing the service invocations and service transitions required by an application that aggregates the user's friends' addresses and projects them on Google Maps. Note that the transitions are labeled with conditions on the returned API calls. The web services composition and choreography described by BPEL can be formalized based using finite state processes (FSP) (Foster et al., 2005; Salaun et al., 2005; Foster et al., 2006; Hogben, 2007). In what follows we define the application as a transition system.

Definition 3. *(Application Transition System). An application transition system is a tuple* $TS = (S, \Sigma, \delta)$*, where:*

- *S is a finite set of states. The set of states includes a single initial state* s_0 *and a finite set of final states* $F \subseteq S$.
- Σ *is the alphabet of operations offered by the service and the data required by this service.*
- $\delta : S \times \Sigma \rightarrow S$ *is the transition function that maps states and alphabets to another state. The transition* $\delta(s_i, \alpha) = s_j$*, represents that transition from state* s_i *to state* s_j *subject to services and data in* α.

The final states $F \subseteq S$ describe the different flavors of a given application, for example, a horoscope application could simply provide a user with her daily horoscope, or her horoscope combined with her friends' horoscope, or her horoscope and compatibility with her friends, etc. The mapping function δ is used to represent the constraints required to transition from one state to another. In this paper, we focus on constraints related to the required profile data generalization levels requested by the application to enable the successful transition from a state to another. For example, an application requesting the user's address through the service *get_address*(), the application will transition to a different state depending on the generalization level of the returned address attribute. From an application perspective the user generalization preference vector specifies the permitted attribute generalization levels, which in turn dictates the set of permissible state transitions. The set of final states represents the different service levels provided by the application.

Definition 4. *Given an application transition system* $TS = (S, \Sigma, \delta)$ *and a user preference vector UP, the reduced application transition system* TS_{UP} *is defined as the tuple* $(S_R, \Sigma_R, \delta_R)$*, where:*

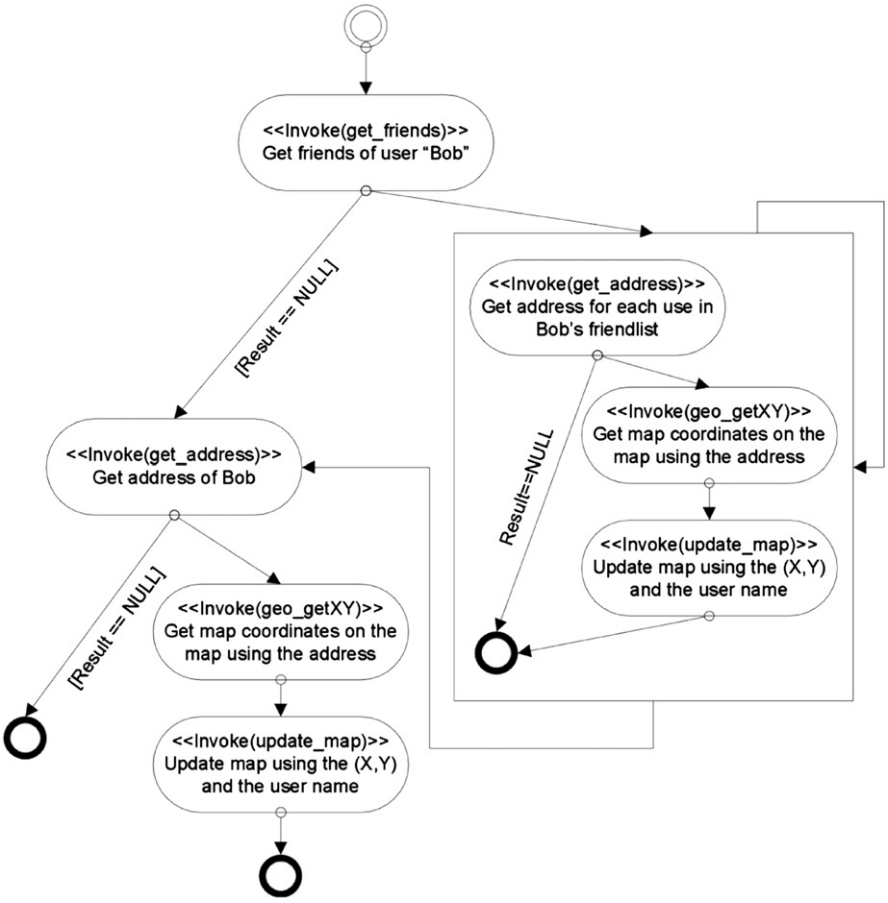

Fig. 6 – Example application process.

- $S_R = S$ and $\Sigma_R = \Sigma$.
- $\delta_R = \delta$ for $\delta(s_i, \alpha) = s_j$

where the attributes α satisfies the user preference vector UP.

The reduced application transition system includes only the state transitions that are permitted by the user preferences. It also indicates the states that are reachable after the user preferences are applied to the application.

We model the application transition system *TS* as a directed graph $G = (V, E)$, where the vertices V represent the states, and the edges E represent the state transitions. The edges E are labeled with the minimum attribute generalization levels required to enable the state transition. For an edge $e \in E$ the edge label *e.h* represents the generalization level required for the state transition. For example, in Fig. 7(a) the edge (S_0, S_1) is labeled with h_2^1 indicating that the generalization level 2 is required for attribute x_1 to enable transition from state S_0 to state S_1. A user preference is said to satisfy a transition if the specified user attribute generalization level is greater than or equal to the edge generalization level. The reduced application transition system is computed by generating a graph $G_R = (V_R, E_R)$, where $V_R = V$ and $E_R \subseteq E$ includes only the transitions E that satisfy the user preferences. Fig. 7(b), shows an example reduced application transition graph for the user preference vector $up = \{h_1^1, h_1^2, h_2^3, h_1^4\}$ and the original application state diagram in Fig. 7(a).

Definition 5. *(Application Service Path) Given an application transition instance TS, the path $P = \{e_0, \ldots, e_{n-1}\}$ is sequence of state transitions, where the edge e_0 starts at the initial state s_0 and the ending edge e_{n-1} terminating at a final state $s_n \in F$. The path generalization vector $g(P) = \{e_1.h, \ldots, e_{n-1}.h\}$ is defined as the set of data attribute generalization levels required to traverse this path.*

The Application Service Path represents an instance of an application execution that starts at the start state s_0 and ends at a target ending state s_n.

5.1. Optimal user application preferences

In our framework, when trying to install an application, the user specifies an attribute generalization preferences and a target final application state. The challenge the user is faced with is to identify the minimal attribute generalization preference required to enable the application to successfully terminate to the requested final state. According to the security principle of *Least Privilege*, an application should be awarded access to the smallest set of profile attributes at the minimum generalization levels in order to provide the requested service. Formally, the minimal attribute generalization problem is defined as follows:

Definition 6. *Minimal Attribute Generalization Problem, Given an application transition instance $TS = (S, \Sigma, \delta)$, and a target final state $s_f \in F$, determine the minimal user attribute vector $UP^* = [h_1^*, \ldots, h_n^*]$ required to enable the successful transition from the start state s_0 to the final state s_f.*

The minimal user attribute vector is the vector that requires the minimum exposure of the user attributes and enables the application to transition to the target final state. Using the graph based application transition model, an application service path beginning at start state and terminating at the final target state holds the set of generalization levels required to take such a path. The minimal attribute generalization problem translates to finding the minimal application service path from the start state to the target final state in a weighted application transition system defined as follows:

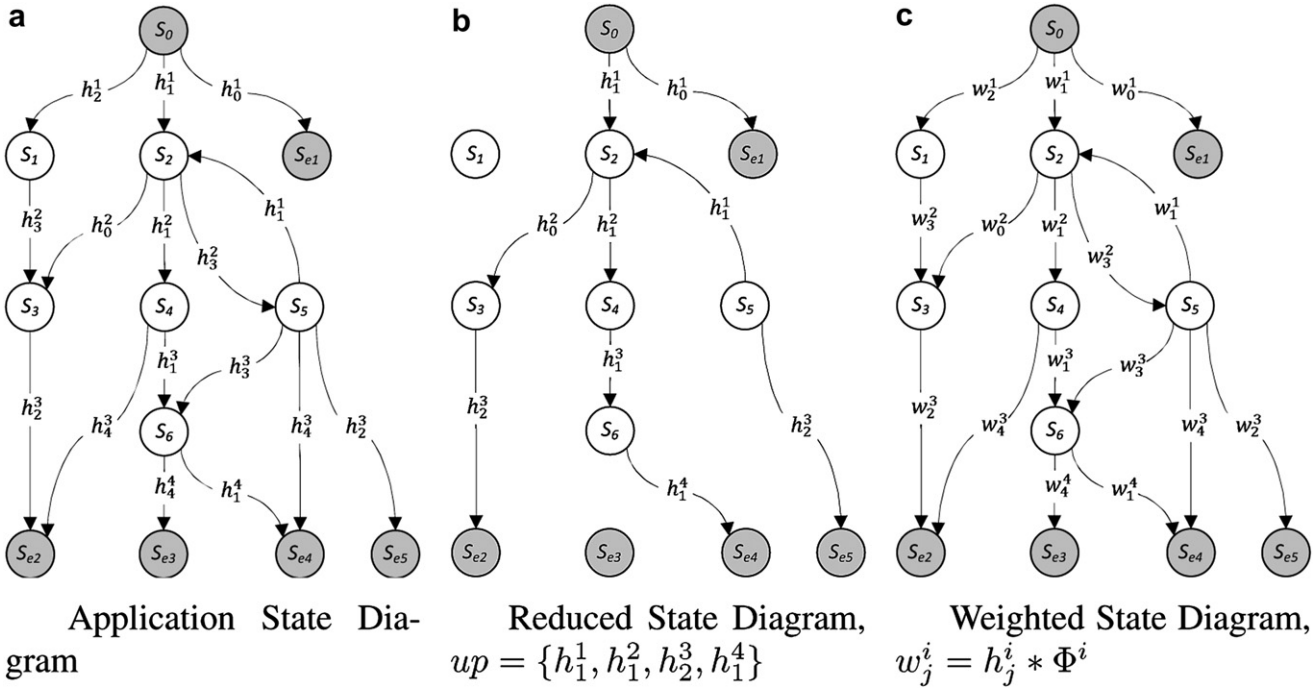

Fig. 7 – Application state diagram and user preferences.

Definition 7. *(Weighted Application Transition System). A weighted application transition system* $TSW = (G, W)$ *where:*

- *G is the application transition graph* $G = (V, E)$*, where V is the set of vertices representing the finite set of states, and E is the set of edges representing the state transitions.*
- $W : E \times \Phi \rightarrow w \in \Re^{+}$ *is the edge weight function that maps the edge attribute generalization labeling E.h and the attribute sensitivity* Φ *to an edge weight w.*

Given an application service path $P = \{e_0, \ldots, e_{n-1}\}$, the path length is defined as follows:

$$\Theta(UP) = \sum_{i=0}^{n-1} W(e_i, \Phi_i) = \sum_{i=0}^{n-1} \Phi_i e_i.h$$

Given the weighted application transition system and the path length definition, the minimal attribute generalization problem simply maps to finding the shortest path from the start state s_0 to the final target state s_f. The initially specified user preferences are used as an upper limit on the user preferences and are referred to as the upper limit user preferences $UPL = [h_0, \ldots, h_n]$. Fig. 8, depicts the algorithm used to compute the minimal user attribute preferences vector. Lines 1–9, initialize the application transition graph to generate the edges that are not allowed by the specified user attribute generalization upper limits buy setting the edge weights to ∞, and the weights of the permitted transitions using the edge weight function that incorporates both the user attribute sensitivity and generalization level. Lines 10–14, initialize the distance from s_0 to other vertices, where $d[u]$ and $pi[u]$ represent the shortest distance from s_0 to u and the predecessor of u on the shortest path respectively. Lines 15–24, computes the shortest path from s_0 to all the transition states. Lines 25–34, computes the minimal user preferences vector required to transition from state s_0 to the target final state s_f.

6. Implementation and experimental results

Our approach to assess the proposed solution is two-fold. First, we investigate the architectural changes that our approach would entail on an existing social network. To this extent, we develop a proof-of-concept implementation using an existing open source framework for social network sites. Second, we

```
Algorithm: generate_minimal_preference
Input: Application transition graph G = (V, E),
User upperlimit preferences UPL = [h_0, ..., h_n], User target state s_t
Output: User Minimal Attribute Preferences UP*

1.   V_R ← V
2.   E_R ← E
3.   //Generating the reduced graph
4.   for each e ∈ E_R
5.     for each h ∈ UPL
6.       if h ≺ e.h
7.           e.w = ∞
8.       else
9.           e.w = Φ_{e.a} * e.h
10.  //Initialize distance from s_0
11.  for each v ∈ V_R
12.    d[v] = ∞
13.    pi[v] = {}
14.  d[s_0] = 0
15.  //Computing Shortest Path from s_0
16.  S = {}
17.  Q ← V_R //Priority Queue on d[u]
18.  while Q is not Empty
19.     u = ExtractMin(Q)
20.     S ← S ∪ {u}
21.    for each v ∈ adjacent(u)
22.      if d[v] > d[u] + w(u, v)
23.          d[v] = d[u] + w[u, v]
24.          pi[v] = u
25.  //Tracing Minimal User Preferences from s_0 to s_t
26.  UP* = {}
27.  if d[s_t] == ∞
28.    return UP*
29.  u = s_t
30.  do
31.     UP* = (pi[u], u).h ∪ UP*
32.     u = pi[u]
33.   while pi[u] ≠ s_0
34.  return UP*
```

Fig. 8 – User minimal attribute preferences algorithm.

show the feasibility of our proposed approach by conducting user studies on a widely-used social network platform.

6.1. System prototype

In this section we present the main extensions we have introduced in order for an application accommodate attributes with different generalization levels based on the user preferences. We begin with a brief presentation of the Drupal platform (Buytaert, 2009), which we used as a platform for our approach, followed by our additions and modifications to the Drupal Site.

Overview of Drupal: Drupal is an open source content management system that is commonly used to build social networks and online communities. Drupal has a modular structure: it offers a number of core and optional modules that can be combined with any modules contributed by the developers, to enable a whole range of functionalities. Drupal's module system is based on the concept of APIs called "hooks". A hook is a function, which has a defined set of parameters and a specified result type. These hooks are used to interconnect and enable communication between the different modules. The modules are also able to store and retrieve data stored in a shared Drupal database. This information is required by the modules to provide their services. Drupal provides a profile module which manages profile information of each registered user in a table called *PROFILE_VALUES*. The columns of the tables are the *uid*, *fid*, and *attr*, which represent the user's unique identifier, the field identifier mapping to the different social network profile fields, and the attribute value respectively. The current Drupal implementation enables all modules to access the user profile information by simply querying the *PROFILE_VALUES* table in the database. The content in Drupal is stored and treated as nodes. Drupal stores the association of the nodes, the contents, the content types and their owners in different tables in a database accessible to all the modules.

Drupal's Extensions. To implement our proof-of-concept prototype, we extended several core modules, including the *profile*, *profile privacy*, *webform* modules. In addition, we developed a Horoscope module to investigate the feasibility of our proposed approach. The horoscope module provides the user with daily horoscope updates. The horoscope being displayed depends on generalization level of the profile attributes that the user is willing to share to the module.

We extended the native profile module to restrict direct access to the profile fields, and to enable access only through the profile module hooks. In this way, the other application modules are able to access the profile fields only through the profile module. The profile module regulates the privacy level of access for each profile information based on the input of the profile owner. The profile module encrypts all the profile information before it is stored into the *PROFILE_VALUES* table in the Drupal database. The encryption and decryption of the profile fields is managed by the profile module. This simple approach ensures that the profile attributes are protected through encryption and only accessible through the profile module hooks. Each application module has different level of access to the profile field information depending on the privacy level specified for this particular application module. The profile module therefore needs to differentiate between the different models and should be able to authenticate the application module before passing on the decrypted profile information to the application module. Fig. 9 illustrates the authentication concept used in this approach.

The application module gives the user the ability to pick the level of operation of a given application module. The application developer is required to specify the minimal profile information that would be required for the different levels of operation of the application module. The profile module stores a local copy of the module name and its corresponding random key, by invoking the *horoscope_enable* and *profile_modules_enabled* hooks. Since *horoscope_enable* and *profile_modules_enabled* are atomic, the key cannot be leaked. Subsequently, to authenticate an application, the profile module checks if the random key passed by an application module to the profile module is identical to its local copy of application module and it's corresponding random key. If they match it authenticates the application module.

Whenever the user enables a particular application module a web form is loaded. Web forms are handled by the Drupal webform module. To enable suggesting customized fields, we modified the process of value retrieval. For example, in Fig. 10(a), we report a partial output of the form generated by the webform of the Horoscope module. Here, the value stored in the "Minimal requirements for the Minimal Service", "Minimal requirements for Intermediate Service-I", "Minimal requirements for Intermediate Service-II" and "Minimal requirement for Maximum Service" are specific to the horoscope module. These values will not be same for all the

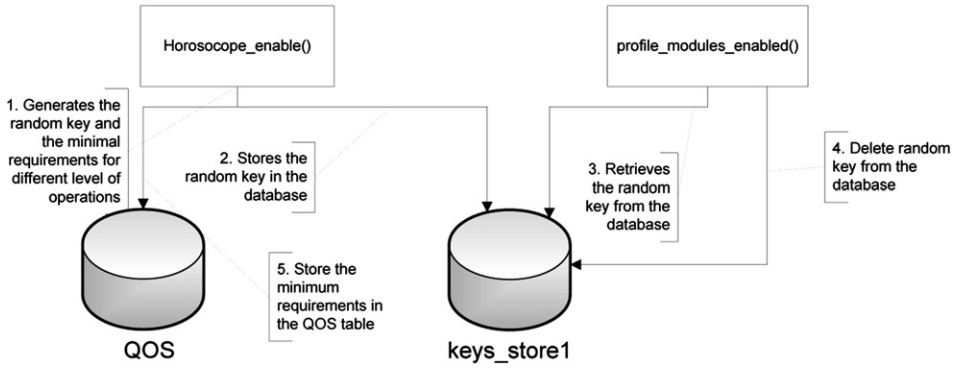

Fig. 9 – Random key generation for authentication and minimal requirements specification.

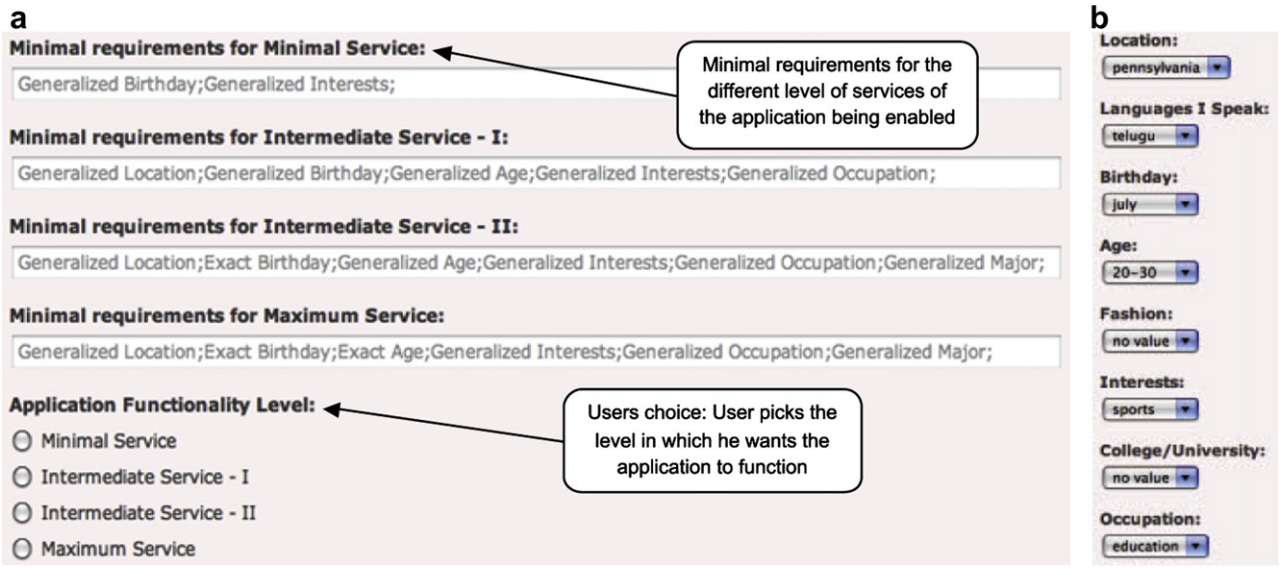

Fig. 10 – (a) User input for application functionality level. (b) User selecting attribute generalization levels for profile data.

application modules, as they are specified through the application sheet. Notice that the form in Fig. 10(a) provides the user with the minimal profile information that would be required by the application module for its different level of operation. The user then picks the level of functionality based on the minimal requirements condition. If the user picks "Intermediate Service-I" but he doesn't meet the requirements for that level, the application will automatically detect the level in which the user can operate based on the profile information that he has selected. For instance, let us assume that the user has selected Generalized Birthday and Generalized Interests as the information to be exposed to the application. He then selects Intermediate Service – I as the Application Functionality Level. In this case, the conditions are not met, but the information provided by the user meets Minimal Service level of the application (in the above figure). Therefore the application operates in Minimal Service instead of the Intermediate Service-I.

Fig. 10(b) shows the generalization levels of each profile attribute, which the user can select for the horoscope example. In the Figure, Pennsylvania, July, and 20–30 represent the generalized values of State College Pennsylvania, 01-July, and age 22 respectively. Furthermore, the user has decided not to expose Fashion and has selected generalized value for Interests which is sports (actual value being tennis). Clearly, the values displayed to the user for generalization are specific to the user who has enabled the horoscope module, and therefore assigned dynamically. These values override the statically assigned initial values of the form generated by the webform module.

6.2. *User studies*

We also evaluated our approach through a user study involving a significant number of participants.[1] Our evaluation involved a simulation of the application installation procedures, followed by a post-session questionnaire, as described in this section.

Participants were randomly recruited from a large US university community (staff, students, and faculty), including users of social network sites. Out of the 250 participants recruited, 85 agreed to participate in our study (response rate of 34%). The average age of the respondents was 25.13 years old. Participants were asked to indicate the social network they most often accessed: 85.6% most often accessed Facebook, while the remaining participants were distributed among Orkut, Linkedin (6%) and LiveJournal. Considering Facebook is one of the social networks that most heavily promotes the usage of applications, our sample was deemed appropriate for our study. In terms of network usage frequency, 94% of the respondents accessed social network sites at least once a week, and of 73.8% of those were daily users. Moreover, in terms of application usage, 47.8% of the respondents reported using applications at least once a week, while 29.8% used applications about once a month. Only 21.7% of the participants never used applications. Most respondents (82.9%) reported having removed applications at times and expressed concerns with installed applications. Participants were also asked to indicate how many minutes they were willing to spend in configuring applications. The responses ranged from 0 to 20 min, with an average of 4.89 min. Below we list the measures utilized for our study.

6.2.1. *Measures*

- Ease of installation using our approach was measured using 3 items ($\alpha = .84$) rated on a Likert scale (5-point rating scale, where 1 = strongly disagree and 5 = strongly agree). An example item is "Installing the last three applications was easy."
- Satisfaction with our approach was measured using 3 items ($\alpha = .71$) rated on a Likert scale. An example item is "I liked the approach used in this study."

[1] Our research study was protected by IRB#30654 "Improving Users' Control Over Third Parties Applications in Social Network Sites".

- Concern with information disclosure was measured using 3 items ($\alpha = .92$) rated on a Likert scale. An example item is "I felt nervous disclosing data to these applications." Concern with information disclosure was measured separately for the applications using the commonly adopted approach and for the applications using our approach.
- Frequency of application use was measured on a frequency rating scale (1 = never to 5 = once or a few times a month) with the item "I use applications in Social Network sites."
- Frequency of access to social network sites was measured on a frequency rating scale (1 = never to 5 = once or a few times a month) with the item "How often do you access online Social Networks sites?"
- General control inclinations were measured using 2 items rated on a Likert scale. The items "I like the idea of controlling my data when installing applications" and "I think it's important to control your profile information and who accesses it" were not significantly intercorrelated ($r = .20, p > .05$). Hence, they were used as individual items and a composite scale was not computed.
- Willingness to reduce application capabilities for security purposes was measured using the item "I would be willing to reduce the application capabilities for better security and privacy," rated on a Likert scale.

6.2.2. *Procedure*

Participants were asked to test our protection mechanism on real social network sites, using a comparative approach. Specifically, participants were asked to login to an existing social network site using their account (we chose Facebook due to its popularity). Participants were then presented with usage scenarios that asked them to simulate the installation process of 3 different applications using two approaches. The first approach entailed installing the 3 applications following the commonly adopted procedure in existing social network sites. That is, participants had no control over the data to be disclosed to the application and could only choose whether to install the APIs using this procedure or not install them at all. The second approach (our approach) entailed installing the 3 applications following our protection procedure. Specifically, participants were allowed to choose the quality of services of each application, taking into consideration the level of information disclosure that was necessary for each quality level. Furthermore, users could generalize the data asked by the application at their own will. The application suggested in a drop-down menu the predefined generalized attributes for each of the attributes required in order for the applications to function. For example, for the attribute location of a user in Pennsylvania, the drop-down menu would show City, State, Country. A screenshot of the installation main page is reported in Fig. 11.

After installing the applications, participants were asked to complete a post-session questionnaire assessing their attitudes toward the two approaches, their experiences with and attitudes toward social network sites and applications in these sites, their general control inclinations, and their demographic characteristics.

6.2.3. *Results*

The main purpose of this study was to examine users' attitudes toward our application installation approach. Table 1 presents the descriptive statistics of the study's variables. We hypothesized that participants would be less concerned disclosing information to applications when using our approach (that allowed them to control the information they would disclose and the information they would generalize), than when using the commonly adopted approach (that does not allow them to control which information they disclose). In order to test our hypothesis, we conducted a repeated measures *t*-test, comparing the mean concern with information disclosure when using our approach to the mean concern with information disclosure when using the commonly adopted approach. The difference in concern was statistically significant ($t = -.37, p < .001$). Participants reported less concern with disclosing information when using our approach ($M = 3.60, SD = .95$), than when using the commonly adopted approach ($M = 3.23, SD = 1.04$). Hence, this result confirms our hypothesis, and proves that users are concerned -as they should be- in disclosing massive amount of personal data to application developers. Moreover, we were interested in understanding the factors affecting users' attitudes toward our approach. Specifically, we examined whether satisfaction with our approach is predicted by participants general control inclinations, willingness to reduce the capabilities of applications for the security purposes, concern with information disclosure when using the commonly adopted approach, concern with information disclosure when using our approach, age, frequency of application use, and ease of application installation. Specifically, we conducted an exploratory least-

Fig. 11 – Screenshot of the application installation.

Table 1 – Descriptive statistics of study variables.

Variable	Mean	Std. Dev.
Ease of installation	4.04	0.71
Satisfaction with our approach	3.96	0.60
Concern with information disclosure when using the commonly adopted approach	3.60	0.95
Concern with information disclosure when using our approach	3.23	1.04
Single items		
I enjoy using applications in Social Network sites.	3.26	(1.23)
I have had concerns about some applications (for example, I have been concerned about my privacy, I have been concerned about my computer security etc.)	3.99	(0.90)
I like the idea of controlling my data when installing applications	4.46	(0.67)
I think its important to control your profile information and who accesses it	4.58	(0.65)
I would be willing to reduce the application capabilities for better security and privacy	4.12	(0.86)

squares multiple regression analysis, regressing satisfaction with our approach simultaneously to all the possible predictors. The most significant predictor of satisfaction with our approach was participants willingness to reduce application capabilities for the sake of security ($\beta = .488, p < .001$). The more willing participants were to reduce application capabilities for the sake of security, the more they liked our approach. Moreover, the more concerned participants were with information disclosure when using the commonly adopted approach, the more they liked our approach ($\beta = .286, p < .05$). Not surprisingly, the more concerned participants were with information disclosure when using our approach, the less they liked our approach ($\beta = -.292, p < .05$). Finally, the easier participants found the application installation process, the more they liked our approach ($\beta = .202, p < .05$). General control inclinations, age, and frequency of application use did not predict participants satisfaction with our approach. Furthermore, we were interested in understanding why participants differed in their perceptions of how easy it was to install applications using our approach. We hypothesized that the younger participants were and the more often they accessed social network sites and used applications, the easier they would find installing applications using our approach. In order to test our hypothesis, we conducted a least squares multiple regression analysis, regressing ease of installation on age, frequency of application use, and frequency of access to social network sites. Participants age was the only significant predictor of ease of installation ($\beta = -.398, p < .001$). Specifically, the younger participants were, the easier they thought installing applications using our approach was. This result is more likely explained by the fact that young generations are more frequent users of social network sites, and spend more time on these platforms in configuring and populating their profiles. However, when singularly considered frequency of social network access and frequency of application usage were not significant predictors of ease of installation.

7. Conclusions

In this paper we have presented an access control framework for social networks developer applications that enables users to specify profile attribute preferences and requires applications to be designed so to be customized based on users' profile preferences. Our framework provided a privacy-enabled solution that is in line with social network ethics of openness, and does not hinder users' opportunities of adding useful and entertaining applications to their profiles. We modeled the applications as finite state machine with transition labeling indicating the generalization level required to enable application state transitions. We defined the reduced application transition system that only includes the state transitions possible with a given user generalization vector. Then we incorporated the user sensitivity metric to generate the weighted applications transition system.

Furthermore, we formalized the Minimal Attribute Generalization Problem and presented the Weighted Application Transition System which incorporates the user attribute sensitivity metric to generated a weighted graph representing the application state transitions. Using the weighted graph we transformed the Minimal Attribute Generalization Problem to the shortest path problem and provided an algorithm that generates the optimal user generalizations vector that will enable the transition to a target final state.

We evaluated the feasibility of our solution by showing a proof-of-concept architecture that extends a widely used open source content management. We showed how, with some extensions to the platform's architecture, it is possible to develop a secure approach limiting the access of users' data to the applications, and disclose only the attributes that the user consented. Additionally, we assessed the users' perceived benefits and the ease of use of this type of approach by conducting a user study. As presented in the paper, the results are positive; users acknowledge that these types of solutions are needed and that our approach would allow them to enjoy more confidently the functionalities offered by applications. In the future, we plan to investigate the current work along several directions. First, we plan on extending the functionalities of the generalization technique, to support dynamic and customized generalization values. We will explore whether ontologies can be integrated in the social network system, so as to support a large variety of generalized values. Also, one limitation of the current prototype, is the lack of control of the data once it is disclosed to one application. An application may still disclose such user's data to others, leaking users' private data. To avoid such information flow issues, we are investigating stronger techniques that could allow a more stringent control over the data disclosed to each application.

REFERENCES

Acquisti Alessandro, Gross Ralph. Imagined communities: awareness, information sharing, and privacy on the facebook. In: Privacy enhancing technologies, pp. 36–58; 2006.

Baden Randolph, Bender Adam, Spring Neil, Bhattacharjee Bobby, Starin Daniel. Persona: an online social network with user-defined privacy. In: SIGCOMM, pp. 135–146; 2009.
Besmer Andrew, Richter Lipford Heather, Shehab Mohamed, Cheek Gorrell. Social applications: exploring a more secure framework. In: Proceedings of the 5th Symposium on Usable Privacy and Security, SOUPS'09. New York, NY, USA: ACM; 2009. p. 2:1–2:10.
CNET Blog. Exclusive: the next Facebook privacy scandal, http://news.cnet.com/8301-13739_3-9854409-46.html; 2008.
Carminati Barbara, Ferrari Elena, Perego Andrea. Rule-based access control for social networks. In: OTM Workshops (2), pp. 1734–1744; 2006.
Carminati Barbara, Ferrari Elena, Perego Andrea. Enforcing access control in web-based social networks. ACM Transactions on Information and System Security (TISSEC) 2009;13(1).
Dries Buytaert. Drupal platform, http://drupal.org/; August 2009.
Damiani E, Vimercati S, Paraboschi S, Samarati P. A fine-grained access control system for XML documents. ACM Transactions on Information and System Security 2002;5(2):169–202.
Facebook Inc. Facebook's privacy policy, https://www.facebook.com/policy.php; Dec 2010.
Facebook Inc. Facebook asks more than 350 million users around the world to personalize their privacy, http://www.facebook.com/press/releases.php?p=133917; December 2009.
Federal Trade Commission (FTC). Facebook settles ftc charges that it deceived consumers by failing to keep privacy promises, http://www.ftc.gov/opa/2011/11/privacysettlement.shtm; Nov 2011.
Felt Adrienne, Evans David. Privacy protection for social networking platforms. In: Workshop on Web 2.0 Security and Privacy; May 2008.
Fong Philip WL. Relationship-based access control: protection model and policy language. In: Proceedings of the first ACM conference on data and application security and privacy, CODASPY'11. New York, NY, USA: ACM; 2011. p. 191–202.
Foster H, Uchitel S, Magee J, Kramer J, Hu M. Using a rigorous approach for engineering web service compositions: a case study. In: 2005 IEEE international conference on services computing, vol. 1, pp. 217–224, July 2005.
Foster H, Uchitel S, Magee J, Kramer J. Ltsa-ws: a tool for model-based verification of web service compositions and choreography. In: Proceeding of the 28th international conference on software engineering, pp. 771–774, May 2006.
Gates Carrie E. Access control requirements for Web 2.0 security and privacy. In: W2SP 2007: Web 2.0 security & privacy, p. 3, 2007.
Geambasu Roxana, Balazinska Magdalena, Gribble Steven D, Levy Henry M. Homeviews: peer-to-peer middleware for personal data sharing applications. In: SIGMOD conference, pp. 235–246; 2007.
Golbeck Jennifer, Hendler James A. Inferring binary trust relationships in web-based social networks. ACM Transactions on Internet Technology 2006;6(4):497–529.
Google Code. Google's developer network, http://code.google.com/; 2009.
Google Inc.. Google+ privacy policy, http://www.google.com/intl/en-US/+/policy/; June 2011.
Gollu Kiran K, Saroiu Stefan, Wolman Alec. A social networking-based access control scheme for personal content. In: Proc. 21st ACM Symposium on Operating Systems Principles (SOSP'07). Work in progress; 2007.
Hart Michael, Johnson Rob, Stent Amanda. More content – less control: access control in the Web 2.0. Web 2.0 Security & Privacy 2003.
Hart M, Johnson R, Stent A. More content – less control: access control in the Web 2.0. In: W2SP 2007: Web 2.0 security & privacy, p. 3; 2007.
Hogben Giles. Security issues and recommendations for online social networks. ENISA Position Paper N.1; 2007.
IEEE. W2SP 2008. Web 2.0 security and privacy; 2008.
Irvine M. Social networking applications can pose security risks. Associated Press; April 2008.
Liu Kun, Terzi Evimaria. A framework for computing the privacy scores of users in online social networks. In: ICDM 2009, The ninth IEEE international conference on data mining, pp. 288–297; December 2009.
Masoumzadeh Amirreza, Joshi James. Ontology-based access control for social network systems. International Journal of Information Privacy, Security and Integrity 2011;1(1):59–78.
MySpace Inc. Myspace privacy policy, http://www.myspace.com/index.cfm?fuseaction=misc.privacy; 2009.
O'Reilly T. What is Web 2.0. O'Reilly Network; September 2005. pp. 169–202 [Online].
OASIS. OASIS WSBPEL TC Webpage, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel; 2008.
Porter Felt Adrienne, Greenwood Kate, Wagner David. The effectiveness of application permissions. In: Proceedings of the 2nd USENIX conference on Web application development, WebApps'11. Berkeley, CA, USA: USENIX Association; 2011. pp. 1–12.
Rizvi S, Mendelzon A, Sudarshan S, Roy P. Extending query rewriting techniques for fine-grained access control. In: SIGMOD'04: proceedings of the 2004 ACM SIGMOD international conference on management of data. New York, NY, USA: ACM; 2004. p. 551–62.
Salaun G, Bordeaux L, Schaerf M. Describing and reasoning on web services using process algebra. In: Proceedings of the IEEE international conference on web services, pp. 43–51; June 2005.
Saltzer J, Schroeder M. The protection of information in computer systems. Proceedings of the IEEE Sept 1975;63(9):1278–308.
Samarati P, Sweeney L. Generalizing data to provide anonymity when disclosing information (abstract). In: In PODS'98: proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems. New York, NY, USA: ACM; 1998. p. 188.
Shehab Mohamed, Squicciarini Anna Cinzia, Ahn Gail-Joon. Beyond user-to-user access control for online social networks. In: ICICS, pp. 174–189; 2008.
Singh Kapil, Bhola Sumeer, Lee Wenke. xbook: redesigning privacy control in social networking platforms. In: Proceedings of 18th USENIX security symposium, pp. 235–246; August 2009.
Sweeney L. k-anonymity: a model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 2002;10(5):557–70.
Tootoonchian Amin, Kumar Gollu Kiran, Saroiu Stefan, Ganjali Yashar, Wolman Alec. Lockr: social access control for Web 2.0. In: WOSP'08: proceedings of the first workshop on online social networks. New York, NY, USA: ACM; 2008. p. 43–8.
Vanetti Marco, Binaghi Elisabetta, Carminati Barbara, Carullo Moreno, Ferrari Elena. Content-based filtering in on-line social networks. In: Proceedings of the international ECML/PKDD conference on privacy and security issues in data mining and machine learning, PSDML'10. Berlin, Heidelberg: Springer-Verlag; 2011. p. 127–40.
Washington Chronicle. Study raises new privacy concerns about Facebook, http://chronicle.com/free/2008/02/1489n.htm; 2008.

Mohamed Shehab is an assistant professor in the Department of Software and Information Systems, College of Computing and Informatics, University of North Carolina at Charlotte. He is the director of the Lab of Information Integration Security and Privacy. His research interests lie in network and information

security, especially in the design and implementation of distributed access-control protocols to cope with the requirements of emerging distributed social networks, mobile applications, web services, and peer-to-peer environments. Shehab received a PhD in computer engineering from Purdue University. He is a member of the IEEE Computer Society and the ACM. Contact him at mshehab@uncc.edu.

Anna Cinzia Squicciarini (M'08) received the Ph.D. degree in computer science from the University of Milan, Milan, Italy, in February 2006. She is currently an Assistant Professor in the College of Information of Information Science and Technology, Pennsylvania State University, University Park, PA. During 2006–2007, she was a Postdoctoral Research Associate at Purdue University, West Lafayette, IN. She is the author or coauthor of more than 40 in refereed journals, and in proceedings of international conferences and symposia. Her research interests include access control for distributed systems, privacy, security for Web 2.0 technologies, and grid computing.

Gail-Joon Ahn, Ph.D, CISSP is an Associate Professor of Computer Science and Engineering Program in the School of Computing, Informatics and Decision Systems Engineering at Arizona State University and Director of Laboratory of Security Engineering for Future Computing (SEFCOM). His research foci include access control, secure information sharing, vulnerability and risk management, identity and privacy management, security-enhanced computing platforms, security architecture for networked distributed systems, and modeling for computer security. He is a recipient of Department of Energy Early Career Principal Investigator. He earned his MS and PhD degrees from George Mason University, Fairfax, Virginia in 1996 and 2000, respectively.

Irini Kokkinou is a Professor of Liberal Arts at the Savannah College of Art and Design (SCAD). Professor Kokkinou's research interests lie in the areas of motivation, self-regulation, creativity, work family conflict, and discrimination in employment. She received a Ph.D. in Industrial/Organizational Psychology from Purdue University. She has taught courses in general psychology, industrial/organizational psychology, and statistics at Purdue University, IUPUI, and SCAD. You may contact Professor Kokkinou at ekokkin@scad.edu.