Social Applications: Exploring A More Secure Framework

Andrew Besmer, Heather Richter Lipford, Mohamed Shehab, Gorrell Cheek
Department of Software and Information Systems
University of North Carolina at Charlotte
9201 University City Blvd., Charlotte, NC 28223
1-(704)-687-8387

{arbesmer, heather.lipford, mshehab, gcheek} @ uncc.edu

ABSTRACT

Online social network sites, such as MySpace, Facebook and others have grown rapidly, with hundreds of millions of active users. A new feature on many sites is social applications – applications and services written by third party developers that provide additional functionality linked to a user's profile. However, current application platforms put users at risk by permitting the disclosure of large amounts of personal information to these applications and their developers. This paper formally abstracts and defines the current access control model applied to these applications, and builds on it to create a more secure framework. We do so in the interest of preserving as much of the current architecture as possible, while seeking to provide a practical balance between security and privacy needs of the users, and the needs of the applications to access users' information. We present a user study of our interface design for setting a user-to-application policy. Our results indicate that the model and interface work for users who are more concerned with their privacy, but we still need to explore alternate means of creating policies for those who are less concerned.

Categories and Subject Descriptors

H.m [**Information Systems**]: Miscellaneous

General Terms

Security, Human Factors

Keywords

Access Control, Privacy, Security, Social Networking Applications, Web 2.0

1. INTRODUCTION

Online social network sites such as Facebook, MySpace, LinkedIn and others are experiencing tremendous user growth, with hundreds of millions of active users. Beyond just creating profiles and connecting with friends, many sites are creating a platform for a variety of applications built on top of users' profiles. Facebook and Google's OpenSocial consortium are leading this effort [6][12]. These social applications will become a new paradigm of online interaction where services utilize users' personal information and social connections.

Social applications enhance the functionality and experience of social networking sites by allowing 3rd party developers to add content to a user's profile or provide new social activities. Popular applications allow users to share photos, play games, and share books they have read and movies watched with friends and acquaintances. In order to provide meaningful and engaging experiences, these applications consume user profile data, e.g., name, birth date, interests, and more. To make the matter more complicated, on current platforms, applications can also consume the profile data of the user's friends. The result of this is that a large amount of user data is now available to third parties. Applications can get access to this data without explicit user consent or knowledge, as the only consent obtained is from those adding the application. To demonstrate this risk, the BBC News developed a malicious application that had the potential to harvest large amounts of user profile data [1] in just three hours. Thus, applications contribute to the growing privacy concerns surrounding user profile data stored in online social networking systems.

Currently, social network sites provide few mechanisms for limiting the exposure of user profile data to applications. Facebook, for example, takes an all-or-nothing approach: when a user visits an application for the first time, they must consent to allow that application to access all allowable profile data. This clearly violates the well known principle of least privilege [14]. The only alternative is to not use or visit the application at all. However, even this does not provide any real protection. The application can still request a users information on behalf of a friend who did install the application.

Previous research proposes to secure user information by almost completely limiting what an application can access [7]. Yet, this may in turn limit the usefulness of social applications. We believe that there exists a balance between the privacy of the user and the social value of applications consuming users' and their friend's data. We are investigating a new model for social network application platforms, and do so in the interest of creating that balance while maintaining as much of the current architecture as possible.

In this paper we first present and formally define the current access control model used by application platforms. We then attempt to improve upon this by introducing a new user-application policy to provide protection while still allowing desirable information access. We then explore users' behaviors in utilizing a potential interface for this new access control model. Finally, we discuss the potential implications for protecting personal information on social application platforms, extensions, and improvements to our current prototype.

2. RELATED WORK

Users of social network sites are sharing an amazing amount of personal information including contact information, political and sexual preferences, photographs, personal associations and more [9][17]. For example, studies at Carnegie Mellon University found that over 70% of students disclosed their picture, birthday,

Symposium On Usable Privacy and Security (SOUPS) 2009, July 15-17, 2009, Mountain View, CA, USA.

hometown, and high school information in their profiles [8] This proliferation of personal data puts individuals at risk for serious physical or online attacks, such as stalking, identity theft, and spear phishing attacks [16]. Traditionally, privacy research in social networks has involved protecting personal information in the form of user-to-user policies [2][5][11]. Applications, however, have almost no such privacy controls. And since users' original intent on social network sites is to share data with their friends and communities, they may overlook the risks associated with sharing data with applications and their developers.

Studies such as one performed by Rabkin [13] demonstrate the risks, showing that many common challenge questions used to reset or remember passwords can be answered automatically by using Facebook profile data. Much of this profile data can be harvested automatically from users by malicious applications. Even seemingly harmless applications, such as a chess game, could be unscrupulously storing an unknowing user's data.

A study by Felt and Evans of 150 of the top applications on Facebook showed that most only needed access to a user name, list of friends, and networks to which they belonged to function correctly [7]. However, current application platforms allow any application to query a much larger set of data, whether needed or not. Thus, as Felt explains, as many as 91% of current social networking applications have access to data they do not need, violating the principle of least privilege [7][14].

In order to preserve user privacy with these applications, researchers have suggested that social networking platforms use a privacy by proxy design [7]. Privacy by proxy allows an application access to the user id number and uses a customized markup language to display user data. For example, the application would only have a tag <name id="123"/>. The social network would then fill in the data value, "Andrew Besmer," on the page rendered to the user. As a result, applications have access to no data fields at all, and none of the user's data is put at risk.

However, this approach severely limits the social value of many applications, such as giving a horoscope, mashing interests with recent news, or comparing how a friends' interests matches yours. In cases where it does allow this it forces the developer to expose their business logic, usually in the form of javascript, to the social network and its users. A significant amount of potential value in the platform is lost as a result of such a strict approach.

Facebook, was the first major social networking site to offer an application platform to its users. They opted to create an API complete with its own querying language, FQL. The platform has been very successful with a huge adoption rate and a large developer base of 400,000 [6]. Facebook reports that as many as 95% of their active users use applications. Google then introduced OpenSocial, an open platform that will be common across a number of popular social network sites such as MySpace, LinkedIn, and many others. OpenSocial's early implementations had more of a privacy by proxy design than Facebook's API [12], offering potentially greater privacy protections. However, more recent implementations have included a REST based API which allows for data to be more portable and used on the hosting server, resulting in a platform very similar to Facebook's.

This portability and sharing of data is becoming even more important as these application platforms will allow social network sites to create identity management systems that allow profile data to be used across other websites. These systems would allow the social network user to manage their identity on one site, and have it reflected at all sites to which they belong. Such functionality would not be provided if the application platform was too restrictive. Yet this will only increase the complexity of privacy and security management as even more user data is consumed and shared across applications and sites.

We feel that the balance between the benefits of social applications and user privacy may lie with another approach, by taking a more granular view of regulating what user profile attributes an application can consume. Our approach seeks to preserve much of the current architectures, and allows for portability and desired information sharing, while still providing additional privacy protections.

3. CURRENT ARCHITECTURES

Given the similarity of OpenSocial to Facebook's API we are able to generalize the current state of the art. The subjects in social network sites are users, and applications act on behalf of users to provide services. Each user i maintains a user profile ($Profile_i$), which is a representation of the user on the social network and includes information such as the user's name, birth date, contact information, emails, education, address, interests, photos, music, videos, blogs and many other attributes. In addition, the profile includes the user's set of friends and installed applications. Current social network architectures have enabled users to specify fine grain access control policies on the profile attributes to control user-to-user interactions. In a user-to-user interaction, the user making a request is referred to as the *viewer* and the requested user's profile is the *target*. The user-to-user interaction is governed by the user-to-user access control policy, $P_{t,v}$. For example, *Bob (*the *target*) specifies a policy outlining what attributes *Alice* (the *viewer*) can see of his personal profile. Many sites let Bob choose whether to share aspects of his information publicly, with *"Only Friends"* or with *"Friends of Friends"*. Facebook provides users with the ability to set policies based on particular groups of friends as well.

Applications can provide services to users and interact with users by combining and aggregating attributes from multiple target profiles. For example, an application that provides birthday gift recommendations for a user's friends needs to access the user's profile and her friends' profiles to access their birth dates and interests to be able to make an appropriate and timely birthday gift recommendation. Social network frameworks provide a set of API's that enable third party applications to interface and access user profile attributes. We refer to the set of attributes accessible through the exposed API'S as S. For example, Facebook currently allows most profile data to be accessed, except contact information such as phone numbers and email addresses. OpenSocial is similar, with individual sites adopting the platform able to reduce certain fields within S in different ways.

Current frameworks adopt an all-or-nothing policy when it comes to applications. In other words, upon installing an application, the application is given access to all the attributes in S. Thus, when a user i installs application App_j, the application

has access to all the attributes in (S) that make up user i's profile. The application can also access target profiles of user i's friends by acting as a viewer on behalf of user i. This introduces the ability for an application to access profile attributes of users that have not installed the application or consented to its use.

When the target users create the user-to-user policy $P_{t,v}$, their intention is that they are granting access to other people, and not to installed applications accessing their profiles through their friends. Thus, this current model treats an application that is not installed by the viewer as one of the viewer's friends, and as the viewer himself if it is installed. This violates users' expectations and provides little protection to users who wish to use any applications. This also makes it very easy for a seemingly useful application to maliciously access large amounts of personal information.

To resolve this issue, current social network frameworks enable users to specify a default access control policy D_t for all applications which the user has not installed. User t specifies the set of profile attributes that applications that are not installed by user t are allowed to access. In Facebook, the user's Name, Networks, and List of Friends are available to all applications. However, a user such as *Bob* may also allow or restrict other profile attributes, e.g., Profile Picture, Education History, Work History, etc. The default policy provided by the sites is usually permissive in order to promote sharing.

After all policies have been taking into consideration we can determine the set of all possible data attributes accessible to an application accessed by a viewer who would be able to make requests for information about a target. This can be described as $R_{App_j,v,t}$, which is restricted by the user-to-user policy, and then additionally restricted for targets who have not installed the application. We have more formally defined this access, a list of attributes, and policies in the Appendix. To improve upon this model, we propose a more granular framework that would further restrict the access granted to applications.

4. A GRANULAR FRAMEWORK

We introduce a social network application access control model that addresses the shortcomings of the current application access control model. Our goal is to maintain as much of the current architecture as possible, as current platforms are already widely used. Facebook, for example, reports having over 24,000 applications by 400,000 developers. Our model improves upon the model presented in the previous section by introducing a user-to-application policy on top the previously described framework. This restricts the application to access only those attributes specified in the user-to-application policy. Our model presents several restrictions on the access granted to the application in order to enforce user preferences and create a more secure execution environment.

4.1 Our Approach

We introduce the user-to-application policy, $A_{App_j,i}$, specified by any user i outlining the specific access restrictions for the application. This additional policy has two effects. First, it restricts what information the application can access for the person who installs it. For example, Alice specifies what attributes application App_1 can access of her profile. Second, the policy also restricts what information the application can request of a user's friends on behalf of a user. We refer to this as *friendship based protection*.

Our model gives the viewer finer control in deciding their exposed attributes to applications. In addition to providing a mechanism to enforce fine grain access control on applications, our proposed model also provides a mechanism for social collaboration in making policy decisions, where the viewer is able to influence the set of target attributes accessible by the application, which is a fundamental difference when compared to the current social network implementations, including Facebook. Given the large set of active users on these social networks it would be advantageous to harness their collective knowledge to allow others to make informed policy decisions.

Just like the existing model, it still uses the user-to-user policy to additionally govern what an application can request on another user's behalf. And, it still includes the notion of a default policy for applications that the user has not installed, so a target user can still restrict all uninstalled applications if desired. So again we define $R_{App_j,v,t}$ as the set of attributes accessible to an application that is installed by the viewer, and is used to access the profile of the target user. If the target user already installed the application then the attributes accessible to the application are restricted not only through friendship based protection but his/her own application policy to ensure an application never is able to supersede a user's privacy decision. If the target has not installed the application then the default target application policy D_t will be used. This is again more formally described in the Appendix.

For example, *Bob* (the target) is a careless user who doesn't pay close attention to protecting his profile privacy and leaves his default application policy to be very permissive such that D_{Bob} is the same as S. *Alice* (the viewer) is *Bob*'s friend, and she installed a horoscope application App_1 which is not installed by *Bob*. *Alice* is security conscious and she setup her application policy $A_{App_1,Alice}$ to allow access to only the Birth Date attributes.

The application will now only be able to access *Bob's* birth date when requested by Alice, and nothing more. *Alice's* awareness does not only protect her but it also protects *Bob's* profile due to the fact that *Alice's* policy $A_{App_1,Alice}$ is incorporated when the application App_1 attempts to access *Bob's* profile.

Friendship based protection allows humans to discern for themselves what the minimum set of attributes should be in order to run a given application. Computers are currently not readily equipped or particularly good at making these decisions. Thus, when *Alice* makes in informed decision for herself that decision is in turn used to help *Bob* and the remaining part of the social network. Again *Bob* is further protected if he has specifically prevented his birth date from being known by Alice. In that case App_1would not be granted access to *Bob's* date of birth.

In our model, friends' (viewers) application policies are incorporated when making access decisions, however the policy is upper bounded by the policies set by the target. Thus the addition of the viewer's policy $A_{App_j,v}$ will only help limit the profile exposure and is upper bounded by the policies set by the target. In no case is it possible for a careless viewer's policy to reduce the effects of a protective target's policy. In other words

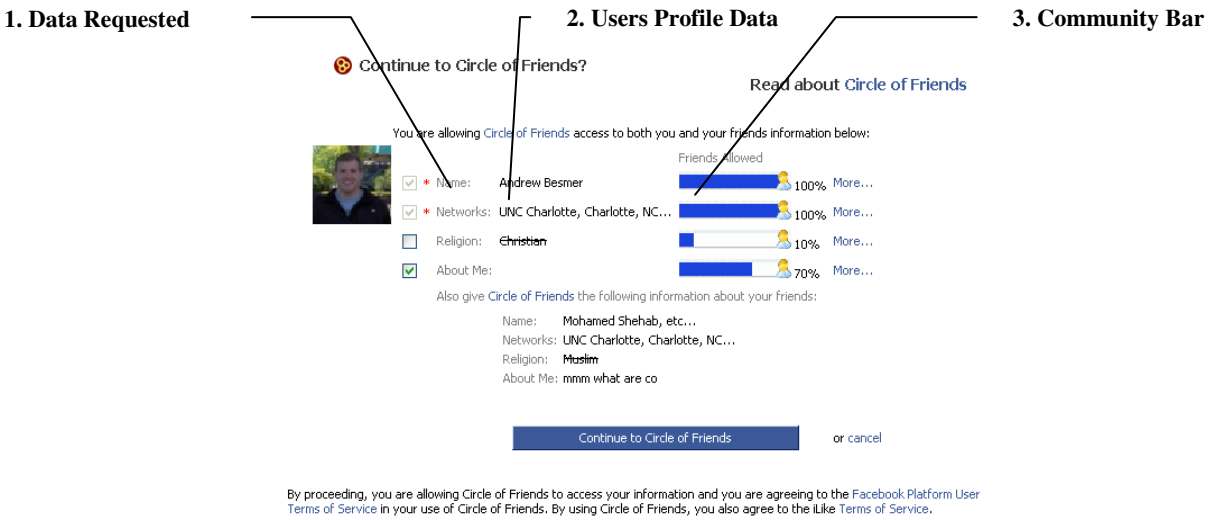

Figure 1 – Prototype User Interface

if *Bob* is security conscious and has set his default policy to what he is comfortable sharing then no decision by *Alice* can reduce the effectiveness of his policy.

4.2 Setting the User to Application Policy

Our access control model requires three different policies to be created: a user-to-user policy, a default application policy, and our new user-to-application policy. Most social network sites already have controls for user-to-user policies, and other researchers including ourselves are focusing on improvements to these controls [11]. So we will not address this policy in our current work. Many sites also already provide a default policy, which users can customize. For example, on Facebook, the default policy settings are one part of the general privacy settings, and users can select which pieces of information they are willing to share or not share with applications they have not installed. Current default policies tend to be permissive to encourage sharing, but can be modified to be very restrictive.

Our model adds a new policy, the user-application policy, $A_{App_j,i}$. The most basic method for creating this policy is to ask users to indicate their choices for each application they interact with. There are several times we could collect this data from users. First, we could ask users to indicate all of their preferences the first time they access or install an application. However, this may take too much time to specify a complex policy. Alternatively, an application could request confirmation for each individual access to a data item, or at least for the first access of that piece of information. However, given user behavior in ignoring frequent popup boxes, such as with invalid certificates on phishing websites [4], we felt this may reduce the likelihood of good privacy decisions and lead to an ineffective solution. Finally, users could have special configuration settings for each application they have accessed to view or modify their policies at any time.

There may be other more advanced methods for setting this user-application policy. For example, policies could allow the generalization of personal attributes [15], e.g., providing state of residence instead of a complete address. In such cases, the user could opt to allow the application to view generalizations of information on their behalf. This generalization might serve to be useful as the user can completely harness applications such as horoscopes and location-based services without exposing the specific data as to their whereabouts or actual date of birth. However, providing this capability creates an even more complicated set of controls and decisions for the user.

Another alternative would be to rely on a voter model, in which more informed users make decisions about the access policies for themselves and for others in the community. Those in the community could then apply a policy based on the number of votes. This would reduce the burden of choosing a policy for a large number of users, yet potentially still provide a useful policy. Of course the down side to such a scheme is that it is subject to gaming, in which several malicious accounts are used to defeat the system.

For our first exploration of our model, we chose to ask users to set all of their preferences upon installation, expanding upon the current screen for providing consent that most if not all platforms already employ. We are currently investigating a basic selection of data fields, without aggregation or voting. With an understanding of how well this method works, we can then explore more advanced methods like these in future prototypes with a better understanding of how users will respond.

4.3 Prototype User Interface

As a step towards understanding and refining our model, we focused our first prototype on the user interface for specifying the user-application policy, and mechanisms to help users choose what information to share or protect. A screenshot of this interface is shown in Figure 1. Users view this interface on installation, the first time they access the application. We focused on several interface features to aid users in making decisions about their policy. First, we allowed applications to request both required and optional information fields. The policy would then automatically restrict any information not requested. When adding or accessing an application for the first time, users would be shown the information that the application requests, and given the opportunity to modify these fields or opt out of the application.

In the interface, we starred the required fields and grayed out checkboxes for those fields, see Figure 1, part 1. We defaulted to all requested fields being selected to encourage sharing, but the user could uncheck any of the optional fields to protect that

information. We also did this in part to preserve the current nature of the applications for users who do not care to protect their privacy. By simply skipping to the continue button users are opting for a full disclosure, such as the one described in the previous model, with little extra time involved. These users however, are still offered greater protection as the application has constrained its own level of full disclosure to those attributes it needs to optimally work. And applications that request extensive amounts of information may get a second look from users seeing so much requested data.

To make each decision more concrete and clear, we added the user's own personal information that will be shared, and information from a randomly chosen friend, to the interface, see Figure 1, part 2. This provides a more accurate mental model of precisely what will or will not be shared with an application. In addition, it serves to catch the user's attention to try and prevent them from skipping this screen even though setting a policy is not their primary task.

We also added a community indicator bar, which is an indication of what percentage of the user's friends have allowed access for this application to each information field, see Figure 1, part 3. This may influence users' decisions regarding which pieces of information to share. Finally, users could read an application-supplied description of how each of the information fields would be used by clicking for more information.

For this prototype, we chose to emulate and implement on the Facebook platform, as that is currently the most active application platform, and Facebook's API allows access to a greater amount of user data than other social networks. This allowed us to more easily recruit users for the user study and provide an extremely realistic user experience.

The wording chosen and buttons to add an application are presented in a similar appearance to Facebook's current application interface. Thus, users click on a "Continue" button to indicate consent and install the application, or "cancel" to leave the application and not set their policy. We have not yet prototyped a method for creating a user-application policy when the user decides not to install the application. One possible action is to immediately lock down the application policy to prevent any data from being shared. This certainly needs to be explored further.

The prototype shown in Figure 1 was implemented as a Facebook application. Thus, users can "install" this prototype on their profile, giving the prototype access to their real profile information to add to the interface. In an actual implementation, the information fields required by an application can be modeled in XML to automatically generate this screen for each application. While we chose a number of real applications for the prototype and user study, we made up which information fields they would want, the explanations, and the values for the community indicator bar.

5. USER STUDY DESIGN

The model proposed relies on user input to regulate the amount of exposure to both the user and the user's friends. We developed a user study to see to what extent users would set policies restricting or allowing that information on Facebook to be released to an application. To accomplish this we prototyped the implementation previously described, which is depicted in Figure 1. We then performed a user study, asking users to go through the process of adding a number of applications to their profile. We told participants that we had created a new application container that we were testing, and that they would be adding applications to their profile using that container, which allows them to limit the information given.

We also informed the participants that they did not have to add any applications they were not comfortable with, and that we would help them remove the ones they did not want to keep at the conclusion of the study. In reality, no applications were added – the prototype was merely a simulation. As a result all our participants were completely protected from malicious applications that might have existed. We did want users to believe they were actually adding the applications in order to make real privacy decisions, and all users appeared to take the task seriously and believe that they were adding these applications to their profile. After the tasks, participants were informed of this and our prototype application was removed from their profile.

All participants were first given a survey which collected demographic information, as well as used Westin's [10] method for categorizing people into privacy fundamentalists, pragmatists, or unconcerned. After participants completed the survey they signed into their own Facebook account and we installed our prototype. They were given no training on the interface.

The prototype presented users with eleven application installation scenarios. All applications were real Facebook applications to enhance realism, but we created the data fields that each application was "requesting" from our participants. Applications we chose may or may not be malicious, as we have no way of truly knowing; this was a major reason for choosing to simulate the experience. We chose a wide variety of functionality including horoscopes, flowers, games, and others. All scenarios were the same for each user, except that the users' own profile information was inserted into the interface. We began with three training scenarios that asked for simple and basic information. The remaining scenarios included two that asked for information that was realistic within the context of the application, two that were unrealistic but seemingly harmless to allow, and two that requested sensitive information that did not fit the applications' context. There was also one application that asked for twenty-eight different attributes and another that did not ask for anything except for name and networks to which the participant belonged.

The realistic scenarios asked for data that made sense given the context of the application. For example, a book recommendation application requested the user's interests and books liked. An unrealistic one was something like a flowers application that asked for political interests. An out-of-context application asked for sensitive information such as a hometown or date of birth, but that request did not fit with the theme of the application and was not justified. Within each category, we presented users with one shorter and one longer scenario of requested information fields.

When participant completed all eleven scenarios, we then interviewed them and asked them several questions including what they liked and disliked about this method of installing applications, who they thought they gave their data to, and feedback on other elements of the interface. We recorded both

the computer screen and audio using usability software for later analysis.

6. USER STUDY RESULTS

We recruited participants through the use of Facebook ads on two campuses and in our city. However due to the requirement to participate in the study on campus, all of our participants ended up being students. We had a total of seventeen participants recruited from two local universities who were active members of Facebook. We had sixteen undergraduate students and one graduate student participate. Eleven of our participants were females and six males. With the Westin survey, we classified ten of these participants as being privacy pragmatists: those that are concerned about privacy but are willing to trade it for benefits. Out of the others, five were classified as being fundamentalists: those who are distrustful of organizations and worry about how their data is used. The remaining one participant was classified as unconcerned.

In general, participants who added applications had two strategies they could use for releasing data to applications. They could use the application's default policy, releasing all required and optional data requested. Or they could also use a custom policy, restricting some or all of the optional information. Some participants were minimalists and unchecked every optional field, others made case-by-case decisions.

Table 1 – Data by Participant

	Privacy	Total Added	Default Policy	Custom Policy	Avg Time
Motivated	**P1**	10	8	2	0:49
	P2	10	3	7	0:22
	P3	7	1	6	0:23
	P4	4	4	0	0:11
	P5	5	4	1	0:26
	P6	11	3	8	0:16
	P7	11	2	9	0:19
	P8	10	3	7	0:21
	Average	**8.50 / 11**	**3.50 / 8.5**	**5.00 / 8.5**	**0:23**
Unmotivated	**P9**	11	11	0	0:16
	P10	11	7	4	0:14
	P11	11	11	0	0:11
	P12	11	11	0	0:09
	P13	5	5	0	0:08
	P14	11	11	0	0:08
	P15	11	11	0	0:05
	P16	3	3	0	0:12
	P17	8	5	3	0:12
	Average	**9.11 / 11**	**8.33 / 9.11**	**0.78 / 9.11**	**0:10**

Table 2 – Comparison of Groups Disclosure

Privacy	Total Added	Default Policy	Custom Policy	Avg Time
Motivated	77.27%	31.82%	45.45%	0:23
Unmotivated	82.83%	75.76%	7.07%	0:10

Table 1 shows the number of applications added, number of times each participant used the strategies described, and average times, for their eleven scenarios. Users generally stuck with one strategy or the other – either accepting all information with little thought, or using custom policies to restrict information or not adding the application at all. We grouped participants by their general strategy, as indicated by the time spent on each scenario, use of custom strategies, and behavior observed on the recordings. We are not attempting to say these groups are statistically different, rather we categorized participants based on observed behavior, both quantitative and qualitative. We refer to these two categories as motivated users and unmotivated users. This categorization was by no means meant to be perfect, but used to illustrate the differences in behavior and discuss our results. Westin's survey had some correlation to these groupings. All fundamentalists fell into the motivated group, along with two pragmatists, and the one unconcerned.

A closer look at three participants labeled P6, P7, and P8 reveals that they used a minimalist strategy. This minimalist strategy shows that they are motivated in restricting applications from acquiring their data. P7 and P8 were both pragmatists and P6 was our sole participant categorized as unconcerned. Others like P13 exhibit behavior that in Table 1 would suggest they belong to the motivated group. P13 however made the decision not to install applications based on if she had seen them before, as she did not want more stuff on her profile. This suggests participants may have also not added applications for social reasons, not just for privacy reasons. Analysis of the recordings left us unclear on P16's intentions. As such we grouped P16 with the unmotivated group based on Westins classification as well as the similar timing data.

The data in Table 2 summarizes the results for the two groups, and shows that there are stark differences between the motivated and unmotivated users. Motivated participants added fewer applications, did not grant applications access to the attributes they requested, set custom policies far more often and on average took more than twice the time to add applications as their unmotivated counter parts.

Specifically, unmotivated participants used whatever the applications asked for 76% of the time they installed, while the motivated users only used that strategy 32% of the time they installed an application. In fact, the unmotivated group only set a default policy 7% of the time. The motivated group instead set a custom policy 45% of the time, or allowed the application access to some subset of requested attributes. The unmotivated group rarely used these strategies. This accounts for the timing difference between the two groups – the motivated group spent time reading and unchecking information fields, while the unmotivated group did not. Table 3 shows the results of these strategies on four of the eleven different scenarios or tasks.

These are representative of two context appropriate and two context inappropriate scenarios. Horoscopes and Books iLike asked for information that seemed somewhat appropriate to the scope of the application. However SpringWater asked for sensitive information, provided no explanation for data it required, and the application's description was written in Hebrew. The HighSchoolBuddies scenario seemed less suspicious, but still asked for several pieces of information that did not fit the application and were not justified.

Table 3 - Comparison of Disclosure Scenarios

	Context Appropriate				Context Inappropriate			
	Horoscope		Books iLike		SpringWater		HighSchoolBuddies	
	Motivated	Unmotivated	Motivated	Unmotivated	Motivated	Unmotivated	Motivated	Unmotivated
Added	100.00%	88.8%	87.5%	88.8%	37.5%	77.7%	62.5%	88.8%
Birthday	*100.00%	*88.8%	-	-	*37.5%	*77.7%	-	-
Hometown	37.5%	66.6%	-	-	*37.5%	*77.7%	-	-
Location	-	-	-	-	0.00%	77.7%	25%	88.8%
Work Info	-	-	-	-	*37.5%	*77.7%	12.5%	88.8%
High School	-	-	-	-	0.00%	77.7%	25%	77.7%
Books	-	-	62.5%	88.8%	-	-	-	-

* Indicates required data to install

For both context appropriate scenarios, the motivated and unmotivated groups had high rates of adding the application and high rates of allowing access to the data. The exception was hometown information for the horoscope application, which might not have made as much sense as we originally planned. The description stated that the hometown information was used to see what stars the participant had been born under. Motivated users usually restricted this information.

SpringWater, a context innapropriate scenario, requested a variety of sensitive information. The motivated group mostly refused to add this application, while the unmotivated group had a startlingly high rate of adding the application and allowing their information to be accessed. However, a signifcantly different number of motivated participants were willing to add the HighSchoolBuddies application as a result of not being forced to release personal information, something that the SpringWater scenario required. Instead, the motivated users restricted the optional fields to protect themselves.

In general the users in the motivated group protected their personal data more than the unmotivated group. Other scenarios had similar results where the unmotivated participants just accepted the defaults. As Table 3 shows, there were, however, still times when the motivated group disclosed personal data to applications which might not have needed it. When reviewing the videos we notice that in some cases motivated users had not entered the data into Facebook, so they might have mistakenly considered themselves protected. For example, if a participant never supplied their favorite books to Facebook they would not consider that option while installing. There is danger in this strategy which we discuss later. In addition, the three participants P6, P7, and P8 used a minimalistic strategy to account for the privacy decisision and may have incorrectly assumed that their information was being protected even though the application may have required sensitive information it did not need.

On applications that asked for data which it didn't seem to need but could be harmless, the motivated group of users used the custom policy to stop that disclosure. For example, in another scenario a flowers application requested participants' political affiliations. Seven of the eight motivated participants added the application and five of those set a policy to remove political affilations from being disclosed. In contrast unmotivated participants who installed the flowers application allowed the disclosure with none restricting it.

To summarize our results, we discovered that participants generally behaved in two different ways. The first group made efforts to minimize the impact of installing an application as well as attempted to make informed decisions about the application's data usage in context. For these users, the interface and the model would work reasonably well. The second group did not, and generally accepted the default policy supplied by the application.

While we desired to evaluate the potential effectiveness of the community bars, there was too little use of custom policies, and thus little impact of the bar in our study. A future study issolating this feature would be needed to determine whether this can persuade users to modify their policies to match their community, and the impact this would have on the overall community.

7. DISCUSSION

The model presented seeks to find a balance between sharing information with applications and protection of user privacy. As such, the model does not offer perfect protection. Our hope is that the results presented here can inform the next iteration of this model and inspire others to do the same. The scope of the problem is enormous as quite literally hundreds of millions of users' personal data could be at risk. We feel users can and still should be able to choose to share information with applications if they desire, and this may lead to inadvertent disclosures. However, the model does potentially reduce the risks by restricting the amount of information disclosed both for individuals using the application and their community of friends. This model is thus highly dependent on the success of setting an appropriate user-application policy. For our first exploration, users were asked to set this policy on installation.

A limitation of our study is that our participants were aware that they were participating in a study, and may have made more restrictive or careful decisions than they would make when faced with installing an application they are really interested in.

Thus, we would need to further verify whether this model would work in deployment. We did try to make the prototype as realistic as possible to have participants believe they were installing these applications.

A number of users were motivated to take the time to review the interface and further restrict the information they shared. As a result, these users reduced their threat, yet still did share some useful information with applications. In addition, they would have added those same protections to their friends and informed the greater community of their choices. This set of users would likely also be motivated to set a restricted default policy, further protecting them from all unknown applications. Thus, we feel that the model and the interface would achieve our desired balance for social applications for those who truly care about their privacy.

However, there are still improvements that could be made. There were still instances of motivated users sharing sensitive data with applications outside a seemingly appropriate context. And while several users were minimalists, and restricted all optional fields, they did still allow sensitive information to be shared when it was required by the application. Thus, minimalists may feel more protected, but could still release their data without full consideration.

Additionally, many users monitor their privacy on their profile by not entering information in the first place, or entering in false information. Thus, the application policy for that data is irrelevant, as it would not impact their real information. However, if users later update their profiles, they may then accidentally disclose that information to applications already installed. This problem is reflected in our current interface as non-disclosed user attributes result in blank spaces within the interface. The next iteration of our model needs to somehow account for these missing attributes. We did observe several cases where a custom policy was set but a personal attribute such as date of birth or hometown was not allowed to be shared with applications or had not been filled in on their profile. In many of these cases users would not make a decision to restrict this attribute.

About half of our participants were not motivated enough to take the time to set their policy or consider whether to add an application in the first place. This unmotivated group left themselves open to disclosures of very personal information, and as a result, also exposed their friends with weak default policies. Arguably the only ones they are putting at risk are others like them who are not going to be willing or do not want to set either the default policy $\mathrm{D_v}$ or the application policy $\mathrm{A_{App_j,v}}$. So the risky users will be hurting other risky users.

Given the population of these social network sites, types of data involved, and ease with which the information can be collected, we find this to be unacceptable. With 200 million active users on Facebook, even if only 20 percent of users engage in risky behavior, that leaves 40 million users subject to greater risks. That is still lot of users and the numbers continue to grow every day. A better social feedback mechanism might be needed to better inform those users when they are making a decision that is contrary to the community's decisions.

There are several reasons users may not be taking the time to set these policies. We believe that users are unlikely to fully understand the risks associated with the disclosure of certain pieces of information, such as their hometown or work history. Thus, they may feel comfortable sharing that information, when perhaps they should not be. This leads us to believe that we either need to increase user motivation to influence more users to more appropriately set their application policies, or reconsider the means in which we set $\mathrm{A_{App_j,v}}$ and $\mathrm{D_v}$. The model presented does not dictate how this is done, and with appropriate policies would effectively preserve the users' privacy and security and promote the social value of applications. This is a tricky balance though because if the policies are too restrictive, some applications will be effectively crippled. If the policies are too open, users who do not change the policies will expose themselves and be the weakest links in their social networks. Thus, more work needs to be done to explore appropriate mechanisms for setting these policies. For example, perhaps policies set by motivated users could be easily adopted or become the default policies for unmotivated users. We intend to explore such variations with future studies where we harness the collective policies of motivated users in some way.

In a much larger sense our results imply that any access control model that relies on users determining their own policies may be subject to similar results. That is, protecting those who are most concerned but inadequately protecting those who are less concerned or less aware of the risks. Perhaps current access control mechanisms need to reexamine the means by which policies are set given the unmotivated groups blatant disregard for their personal data.

It is also possible but less likely in our model for a user-to-user policy $P_{t,v}$ to be violated by an application as a result of this collective information, such as a recent incident involving the TopFriends application [3] on Facebook. The application unintentionally allowed a user to see profile information of others that they should not have access to, because the application cached user data $R^{Eff}_{App_j,t}$. This can be completely mitigated for a given target in our model by a strong default policy, and if they install the application, a strong policy on that application. In the previous model users who have installed the application, even if briefly, have no way of mitigating this potential attack.

8. CONCLUSIONS

We have presented a new model for access control for applications on social network sites. This model adds a new user-application policy which greatly restricts the information applications can access, while still allowing useful and desired information sharing. The model also adds few changes to existing application platforms, enabling our approach to be easily adopted. The success of our model, however, depends upon appropriately setting the new user-application policy. Our first prototype explored having users set this policy upon installation.

We found that for those who were motivated to protect their information, they were able to easily use our interface to set their policy and protect much of their personal information. These users would be able to reap the benefits of social applications and remain reasonably protected against the threats posed by them. In fact, a user could even be using a malicious application without exposing themself to any risk. However, this interface did not work for half our users, who were still unmotivated to protect themselves, and in turn, their friends. For

these users, we need to explore alternate means to set their application policies.

Despite our mixed user study results, we believe the overall model is favorable for several reasons. First, it lessens the likelihood that someone who is concerned about an application accessing his or her data will accidently disclose that information. Second, it prevents the ability of applications that are poorly coded from being exploited by hackers to gain the entire set of a user's data. It also reduces the ability of an application to record all the user attributes and accidently disclose them to others where it might violate the user-to-user $P_{t,v}$ policy, as was done recently by a popular Facebook application. For those who are very concerned, data is not ever released without their explicit consent. And those who are less concerned can opt to release everything just as they are able to in the current architectures.

We will continue investigating the issues raised by our model and user study. In addition, we will seek other ways of setting the user-to-application policy that will improve upon the results of our current prototype and further protect even greater numbers of users.

9. ACKNOWLEDGEMENTS

We would like to thank Gabrielle Bankston who assisted in data collection. The research of Mohamed Shehab has been supported in part by the National Science Foundation (NSF-CNS-0831360) and National Security Agency (NSA H98230-07-1-0231).

10. REFERENCES

[1] BBC News http://news.bbc.co.uk/2/hi/programmes/click_online/7375772.stm, accessed September 29, 2008.

[2] boyd, d. Friendster and publically articulated social networking. In the Extended Abstracts of the Conference on Human Factors and Computing Systems (CHI 2004). Vienna, Austria, 2004, pp1279-1282.

[3] CNet News, http://news.cnet.com/8301-10784_3-9977762-7.html, Accessed September 29, 2008.

[4] Dhamija R., Tygar J. D., Hearst M., Why phishing works, Proceedings of the SIGCHI conference on Human Factors in computing systems, April 22-27, 2006, Montréal, Québec, Canada

[5] Donath J. and boyd d., Public displays of connection. BT Technology Journal, 22:71–82, 2004.

[6] Facebook http://www.facebook.com/press/info.php?statistics, accessed September 29, 2008.

[7] Felt A. and Evans D., Privacy Protection for Social Networking Platforms. In Web 2.0 Security and Privacy 2008, May 2008.

[8] Gross R. and Acquisiti A., Information Relevation and Privacy in Online Social Networks. In Workshop on Privacy in the Electronic Society, 2005.

[9] Jones H., Soltren J., Facebook: Threats to Privacy. MIT, December 14, 2005. Retrieved from http://www-swiss.ai.mit.edu/6805/student-papers/fall05-papers/facebook.pdf.

[10] Kumaraguru P. and Cranor L.. 2005, Privacy Indexes: A Survey of Westin's Studies, ISRI Technical Report, CMU-ISRI-05-138, 2005.

[11] Lipford H., Besmer A., and Watson J., Understanding privacy settings in facebook with an audience view, UPSEC 2008, Berkeley, CA, April 2008.

[12] OpenSocial http://code.google.com/apis/opensocial/, accessed September 29, 2008.

[13] Rabkin A., Personal knowledge questions for fallback authentication. In Symp. on Usable Privacy and Security (SOUPS'08), Pittsburgh, PA, USA, July 2008.

[14] Saltzer J., Schroeder M., The Protection of Information in Computer Systems. Proceedings of the IEEE 63(9), 1278–1308 1975.

[15] Shehab M., Squicciarini A. and Ahn G., Beyond User-to-User Access Control for Online Social Networks, ICICS 2008, October, 2008, Birmingham, UK.

[16] Sophos.com (2007). Facebook ID probe shows 41% of users happy to reveal all to potential identity thieves. Accessed August 8, 2007.

[17] Stutzman F., An evaluation of identity-sharing behavior in social network communities. In the Proceedings of iDMAa and IMS Code Conference, 2005.

APPENDIX

11.1 Current Architectures

The current architecture can be formalized as the following:

- $R_{App_j,v,t}$: The profile attributes for target user t accessible to application App_j through viewer user v.

$$R_{App_j,v,t} = \begin{cases} P_{t,v} \cap S, & t.App_j = 1 \\ P_{t,v} \cap D_t \cap S, & t.App_j = 0 \end{cases}$$

Where:

- $t.App_j$: Is a binary attribute, which is set to true if user t installed application App_j and false otherwise.
- S : Set of attributes available to the application through the api's exposed by the social network.
- $P_{t,v}$: Is the user-to-user profile policy for target user t and viewer user v. *Bob (*the *target*) specifies a policy outlining what attributes *Alice* (the *viewer*) can see of his personal profile. Note, that if viewer and target are the same user, then $P_{t,t} = S$ which is effectively everything provided by the API framework. Most sites allow users to define this policy with respect to public, friends, or even groups of friends.
- D_t : (Default application policy), the set of profile attributes specified by the target user t regarding what applications that are not installed by user t are allowed to access. In Facebook, the user's Name, Networks, and List of Friends are available to all applications. Users can chose to share or protect many other attributes such as interests, hobbies, work, or location.

11.2 Our Approach

Our architecture can be formalized as the following:

- $R_{App_j,v,t}$: The profile attributes for target user t accessible to application App_j through viewer user v.

$$R_{App_j,v,t} = \begin{cases} P_{t,v} \cap A_{App_j,v} \cap A_{App_j,t} \cap S, & t.App_j = 1 \\ P_{t,v} \cap A_{App_j,v} \cap D_t \cap S, & t.App_j = 0 \end{cases}$$

Where:

- $t.App_j$: Is a binary attribute, which is set to true if user t installed application App_j and false otherwise.
- S : Set of attributes available to the application through the api's exposed by the social network.
- $P_{t,v}$: Is the user-to-user profile policy for target user t and viewer user v. *Bob (*the *target*) specifies a policy outlining what attributes *Alice* (the *viewer*) can see of his personal profile. Note, that if viewer and target are the same user, then $P_{t,t} = S$ which is effectively everything provided by the API framework. Most sites allow users to define this policy with respect to public, friends, or even groups of friends.
- D_t : (Default application policy), the set of profile attributes specified by the target user t regarding what applications that are not installed by user t are allowed to access. In Facebook, the user's Name, Networks, and List of Friends are available to all applications. Users can chose to share or protect many other attributes such as interests, hobbies, work, or location.
- $A_{App_j,i}$: The user-application policy is the policy specified by any user i outlining the specific access restrictions for the application to the user's profile attributes.

The effective set of attributes accessible by a given application can be formalized as:

$$R^{Eff}_{App_j,t} = \begin{cases} \left(\bigcup_{v \in V} P_{t,v}\right) \cap S, & t.App_j = 1 \\ \left(\bigcup_{v \in V} P_{t,v}\right) \cap D_t \cap S, & t.App_j = 0 \end{cases}$$

The term $\left(\bigcup_{v \in V}(P_{t,v})\right)$ represents the effective set of attributes contributed by treating the application as a normal viewer and adopting the target to viewer profile policies. Note, that in this model assumes that the application access is controlled solely by the user to user policy.

The effective set of attributes accessible by a given application can be formalized as:

$$R^{Eff}_{App_j,t} = \begin{cases} \left(\bigcup_{v \in V} P_{t,v} \cap A_{App_j,v}\right) \cap A_{App_j,t} \cap S, & t.App_j = 1 \\ \left(\bigcup_{v \in V} P_{t,v} \cap A_{App_j,v}\right) \cap D_t \cap S, & t.App_j = 0 \end{cases}$$