Inference Attacks by Third-Party Extensions to Social Network Systems

Seyed Hossein Ahmadinejad* Mohd Anwar† Philip W. L. Fong*
*Department of Computer Science
University of Calgary
Calgary, Alberta, Canada
{ shahmadi, pwlfong }@ucalgary.ca
†School of Information Sciences
University of Pittsburgh
Pittsburgh, PA, USA
manwar@pitt.edu

***Abstract*—We study inference attacks that can be launched via the extension API of Facebook. We explain the threat of these attacks through a reduction to authentication attacks, devise a taxonomy for such attacks, and propose a risk metric to help subscribers of third-party applications refine their privacy expectations.**

I. Introduction

In the 2006 F8 developer conference, Facebook (FB) announced a repositioning of its business model. Discerning that its core business is the construction of the social graph rather than the development of individual social applications (e.g., photo sharing, blogging, etc), Facebook released its development platform, which allows third-party developers to create social applications that are embedded in the Facebook environment.

Third-party Facebook applications run on their own servers[1]. When Facebook receives a user request to interact with a third-party application, it delegates the request to the application's server. The application subsequently invokes, through a remote invocation API published by Facebook, primitive web services offered by the Facebook server, so as to access the social graph, query user information, contact the user's friends, etc. Application output is sent back to the Facebook server. The latter embeds the received output into the HTML pages that represent the user interface of Facebook. In short, a Facebook application is a remotely hosted software component that augments the feature set of Facebook.

This extensible architecture was then quickly adopted by other social computing platforms. For example, Google's OpenSocial is a cross-platform API with buy-in from a vast variety of social computing platforms, including LinkedIn, MySpace, Ning and Yahoo. Third-party applications developed for the OpenSocial API can be deployed on any social computing platform that supports the API. Although extensibility is a general phenomenon, we anchor our work on Facebook.

[1]The Facebook platform API also supports the development of client-side applications, but they are not the focus of this work.

Extensibility gives rise to new security and privacy challenges: *Since third-party applications are not hosted on Facebook.com, the latter has no way of ensuring that the applications are benign.* The recent investigative report by Wall Street Journal on third-party applications is an illustration of the privacy risk involved [1]. To prevent third-party applications from exploiting the FB platform API for the purpose of harvesting personal information, FB allows users to selectively grant permissions to applications, thereby restricting the latter to access only data required for the normal operation of the applications. This permission-based authorization scheme was refined recently in May 2010, to improve the granularly of control.

The goal of this work is to highlight that such a permission-based authorization scheme is not sufficient (although it is indeed a good baseline protection mechanism). We do so by exploring one particular class of attacks enabled by the extension API of a Social Network System (SNS), namely, the class of ***SNS API inference attacks***. We define an SNS API inference attack to be *the inference of private information by a third-party application through analyzing public information made available by the extension API of an SNS*. Two points are worth emphasizing. First, the user willingly consent to giving access of certain personal data to the third-party application. The mere availability of such ***public data*** does not constitute an inference attack. Second, the user explicitly denies the third-party application of access to certain ***private data***. Yet, by cross-referencing, data mining, parsing, or applying other forms of data analysis to the public data, the third-party application can successfully infer the private data that the user consciously attempts to protect. Permission-based authorization separates public data from private one, but it does not eliminate the correlation between the two.

Compared to other forms of inference attacks [2], [3], [4], [5], [6], SNS API inference attacks are unique in at least three ways. First, limited by permission-based authorization, the attacking application processes public data structures like a "crawler" (cf. [4]). The application has neither global knowledge of the entire social graph nor simultaneous access to the

public profile data of all users. Second, a significant amount of information about a user is authored by parties other than the user. This includes meta data (e.g., time stamp) injected by the platform, and contents about an individual that are contributed by other users (e.g., photo tags). The user has no control over the generation of such data. Third, due to the nature of social computing, a popular third-party application has wide deployment. Viral propagation may lead to mass harvesting of public information. Consequently, the inference engine of a malicious application does not even need to be accurate. A modest probability of success already makes the attack highly dangerous.

The goal of this work is to improve our understanding of SNS API inference attacks. Our contributions are threefold:

1) We assess the threat of SNS API inference attacks by describing how they can be used as building blocks of an authentication attack.
2) We present a taxonomy for classifying SNS API inference attacks. Such a taxonomy allows future work to develop targeted protection mechanisms for each attack variant.
3) We propose a metric for quantifying the risk of SNS API inference attacks. Such a metric could be used for providing feedback to SNS users regarding the adequacy of their privacy settings.

II. Related Work

Inference of private information using social network data has been studied in the past. He *et al.* [2] employ Bayesian networks to infer private attributes of a victim from the public attributes of the victim's friends and friends of friends. Xu *et al.* [3] study the inference of private information from social group membership, again using Bayesian networks. Zheleva and Getoor [5] study eight inference attacks in which users' private information is inferred from public friend lists and public group membership information. The common insight behind the above works is that related individuals, or individuals belonging to the same social group, tend to share common attributes. Sophisticated Bayes classifiers designed particularly for inference attacks on social network data have also been studied [6]. All the previous works above assume that the entire social network data set is made available to the attacker; our work instead studies inference attacks launched specifically from third-party applications. Constrained by permission-based authorization and the platform API, these applications assume the posture of a crawler. The platform API also exposes foreign data authored by parties other than the user, thus creating more opportunities for inference.

Perhaps the most comparable to our work is the work of Bilge *et al.* [4], who study the use of automatic crawlers to gather users' profile information for the purpose of launching profile cloning attacks. Once enough personal information is harvested, the attacker can clone the profile of the victim, either within the same SNS, or in an SNS in which the victim is not registered. Armed with the cloned profile, the attacker now attempts to befriend the friends of the victim. Empirical data suggests that the victim's friends will likely consent to the forging of friendship, thus granting the attacker access to their personal information. Both our work and theirs attempt to gauge the impact of inference attacks through some form of operationalization: gaining the trust of the victim's friends in [4], and subverting authentication in ours. Our work is unique, however, in that third-party applications are controlled by a different access control model than a user-mimicking crawler [4]. The data made available in the two contexts are also different.

A small number of previous work deal with the redesign of an SNS API to reduce the leak of personal information. In [7], a redesign of the permission-based authorization scheme of Facebook is proposed to allow an application subscriber to protect her friends. In privacy-by-proxy [8], the API releases handles of personal data rather than the data itself to prevent information leakage. This present work address the problem of SNS API inference attacks by providing a risk metric to refine the privacy expectation of users.

III. The Threat of Dossier Inference

Solove coined the term ***digital dossier*** to refer to the vast amount of personal data that one surrenders to public records of various forms [9]. The personal information stored in an SNS can be collected and analyzed to construct a relatively complete digital dossier of a user. Most previous works concern the erosion of privacy due to inference attacks on SNSs. Yet, deeper questions are: Exactly what have we lost due to the inference of a digital dossier? How do we measure the damage caused by inference attacks? Unfortunately, the loss of privacy is difficult (if not impossible) to measure. Counting the number of bits of information leaked simply does not suffice.

We propose an operational means to appraise the threat of a hard-to-appraise attack, by reducing it to another threat that we have a much clearer understanding. Specifically, the reduction goes like this: if we can launch an attack $\mathcal{H}$, a hard-to-appraise attack, then it would allow us to use it as a building block to further launch an attack $\mathcal{E}$, a threat we can easily appraise. $\mathcal{E}$ could either cause immediate damages, or erode the effectiveness of a protection mechanism. The possibility to launch attack $\mathcal{E}$ provides us with a clear assessment of what a user could potentially lose in case $\mathcal{H}$ is possible.

In our case, $\mathcal{H}$ is the SNS API inference attack, and $\mathcal{E}$ is an authentication attack. Specifically, in many online services, including financial services, the service provider will negotiate with the user a set of authentication questions that would be used for authenticating the user in case the primary authentication mechanism breaks down (e.g., the user forgets her password), or when extra protection is required (e.g., when the user is logging on from an unusual IP address). When applicable, the service provider will challenge the user with one or more of these questions. If the user provides an answer that matches the one previously agreed upon between the user and the service provider, then the service provider accepts the identity claim of the user, even though other authentication

	Question Category	*Freq.*
1.	**Relationships**: E.g., "*What is your maternal grandfather's first name?*"	16
2.	**Favourites**: E.g., "*What is your favourite hobby?*"	15
3.	**Educational Experiences**: E.g., "*What is the name of the post secondary institution that you attended?*"	4
4.	**First-time Experiences**: E.g., "*What is the name of your first employer?*"	4
5.	**Significant Persons in Significant Events**: E.g., "*What is the first name of the best man at your wedding?*"	2
6.	**Date of Significant Events**: E.g., "*When is your wedding anniversary?*"	1
7.	**Location of Significant Events**: E.g., "*What is the name of the hospital in which you were born?*"	2
8.	**Period-specific Information**: E.g., "*What is the first name of your favourite teacher in final year of high school?*"	3
9.	**Other**	3
	Total	50

Fig. 1. Authentication question types for $\mathcal{F}$.

means failed (e.g., loss of password, dubious location). The assumption is that only the user knows the answers to the authentication questions. Unfortunately, many commonly used authentication questions have answers that can be readily inferred from casual personal information that a user would publish in an SNS. The goal of this section is to demonstrate that the successful launching of SNS API inference attacks could allow an attacker to infer personal information that corresponds to answers of common authentication questions.

To identify the kind of questions commonly used for authentication, we analyzed the authentication mechanism of a real financial institution. We eschew identifying the financial institution, as well as full details of its authentication questions, so as to protect the institution as well as its customers. Let us refer to this institution as $\mathcal{F}$. When a customer opens an account with $\mathcal{F}$, the customer will be asked to select one question from each of the five groups of ten pre-selected authentication questions, for a total of five questions out of fifty pre-selected ones. The customer will provide answers to those questions. When the customer logs in from a location different from the usual one, one of the five questions (selected in random) will be directed to the user. This extra authentication is performed on top of the existing password-based authentication mechanism. Figure 1 tabulates the different categories of questions as well as their frequencies.

Question categories 1 and 2, relationships and favourites, have answers that are occasionally stored directly in an SNS, although such information may be protected by permissions: e.g., a user may choose to make her friend list inaccessible, or to hide her favourites. If an attacker could infer these two categories of information, then 62% of all the authentication questions would have been covered. The next popular categories of questions are 3–7, which cover 26% of the authentication questions. They are questions regarding personal experiences, including educational experiences, first-time experiences, or other significant life experiences such as birth and wedding. Together Categories 1–7 cover 88% of the questions.

An effective attacker targets her inference efforts in questions from categories 1–7. We claim that simple inference attacks exist for many questions in these categories. We outline two to illustrate the techniques involved. In each case, an FB application was developed to empirically confirm the exact permissions needed for the data sources to be accessible.

Example 1. *Suppose the goal of the attacker is to identify the birthday of a user. This information belongs to Category 6 (Date of Significant Events). Suppose further that the user subscribes to a 3rd-party application developed by the attacker. The user denies the application of the* `user_birthday` *permission, thereby making her birthday private, but grants the* `read_stream` *permission so that the application can access the so called Stream data structure (i.e., public). The Stream is a table accessible via the FB Platform API. It stores wall messages, newsfeed, notes and status updates of the user. Once subscription is complete, the attacking application downloads the Stream entries of the user for at least one year. It then scans through all Stream entries looking for wall messages containing the standard birthday wish "happy birthday" (or variants), but not containing the word "belated" (or variants). The FB platform attaches the date of communication to each message. Each occurrence of the standard birthday wish constitutes evidence that the date of communication is the birthday of the user. The attacker application then uses frequency of occurrences to select the most plausible day.*

We do not claim that the above is the most accurate birthday miner. Instead, our goal is to make two points. First, we want to identify the permissions that are needed in order to infer the user's birthday. We want to highlight the fact that even if the user makes her birthday private, making a seemingly unrelated data structure accessible is sufficient in exposing her birthday. Second, we want to highlight the fact that even a simple-minded mining algorithm would already allow the attacker to have modest success in inference. With a wide-scale launch of such an attack, even a modest accuracy could lead to a significant number of successful attacks. For example, say that the birthday miner can make a correct guess 10% of the time, and that the application has an installation base of 50,000 users. We are already looking at 5,000 victims.

Example 2. *The goal is to identify the best man of the user (i.e., Category 5: Significant Person in Significant Events). The attacking application is granted the* `user_photos` *permission (i.e., to access the user's photo albums). Each album and each photo are associated with some user-provided textual descriptions. The user may optionally "tag" the individuals appearing in each photo. After downloading these two pieces of information for the user photos, a mining script scans the photo albums to look for wedding albums, and in these albums, look for photos with textual descriptions containing the keyword "best man" (or variants). Then the miner look for occurrences of names in the descriptive texts, or names in tags of these photos. Each co-occurrence of the keyword "best man" with a name or a tag signifies the possibility that the*

named individual is the best man for the user. The miner than makes a best guess using frequency counts.

Note that the first example can be adapted for inferring other celebrated days (e.g., wedding anniversary), while the second is not specific to "best man": it can be employed to infer other relationships (Category 1).

IV. A Taxonomy of Attacks

In this section, we propose a taxonomy for SNS API inference attacks. Doing so allows us to gain a deeper understanding of how inference attacks arise from the context of extensible SNSs, and thus allows us to design protection technologies that target specific classes of attacks. There are in principle at least two ways to classify SNS API inference attacks. One is by the mechanism through which data is obtained by the attacker to form the basis of inference, and the other is by the type of inference performed by the attacker. In this work we take the first approach for the following reasons. First, as our goal is to better articulate the protection requirements of extensible SNSs, it is more appropriate to examine the space of attacks from the perspective of mechanisms that cause information to be leaked. This would allow future work to consider how the data sources can be better protected. Second, classification schemes that are based on inference techniques tend to be highly generic, and thus applicable to inference attacks in general. This could lead us to lose sight of the specific protection challenges caused by the extension API and the peculiar data structures found in SNSs.

We classify the means of information leakage along two dimensions. The first classification dimension concerns the channel through which information is leaked to a third-party application. This leads to five classes of basic attacks. As we shall see, the five classes are not necessarily orthogonal, for an attack could very well be enabled by multiple channels of information leakage. The second dimension classifies the victim of the attack. This dimension produces three variants for each of the five classes above.

A. First Dimension: Channel

1) Self Disclosure: Private information is directly exposed by a user in a public (usually textual) data field or data stream (e.g., status, wall posts, comments, descriptive texts in photo albums). An example is to post a message on your own "Wall", declaring "My birthday is May 4." Even if the user denies the application to read the birthday field of her profile, the application may still learn about the fact by monitoring other public channels (i.e., the data structures for which the application has been granted access). Such attacks are partly caused by the user's lack of diligence in protecting her personal information.

2) Metadata Association: The SNS automatically injects metadata (e.g., date of creation, author) into various data streams. By making these data streams public, the user also makes the associated metadata accessible to a third-party application. The application can now infer private information entailed by that association. Example 1 is a classical example of this kind of exploits. Because the extension platform annotates each message by its date of communication, the association between this date and the textual content (i.e., birthday wish), becomes a public information that can be leveraged by an attacker for inference. Unlike a Self Disclosure attack, which is due to the voluntary disclosure of private information in public channels, the user has no control over the injection of metadata.

3) Foreign Contents: In an SNS, not all data owned by a user is authored by that user. For example, others can comment on a photo you post. These comments are owned by you (in Discretionary Access Control, the owner of a piece of data is the one who can determine the accessibility of that data), but they are not authored by you. In short, others are allowed to produce contents about a user, but that user is burdened with the responsibility to control access to such foreign contents. In Example 1, the wall messages are sent by others. Consequently, the birthday attack is caused by the association of metadata to foreign contents. Another example of foreign contents is photo tags.

4) Relational Join: In the relational database model, tables can be ***joined*** by associating common entries. For example, if there is a table that represents the mapping from photo IDs to tag IDs, and another table that represents the mapping from tag IDs to the individuals being tagged, then we can construct a table (the join of the two) that associates a photo entry to the individuals tagged in that photo. By making such unique identifiers available (e.g., photo IDs), the extension platform grants an attacker the ability to compute the relational join of two tables, and thus enables certain inferences that would otherwise be impossible. A classical example is the best man inference in Example 2. By computing the join identified above, an attacker can now infer the best man of the user.

5) Frequency Observation: Because a malicious application has access to not only a few entries of a data stream, but the full history of data (e.g., all the historical newsfeeds, all the photo albums, etc), it is possible for the application to compute frequency information from such a vast amount of information. Correlation of various kind can therefore be inferred.

B. Second Dimension: Victim

The second dimension of classification brings into the picture the social graph, the defining data structure of an SNS. The above five attacks can be launched to infer information for three kinds of victims.

1) Subscriber: The first victim is obviously the subscriber of the application. For example, if John subscribes to a malicious application, which in turn infers John's birthday, then the attack belongs to this class.

2) Friend: According to the Facebook Privacy Policy, if Bob is a friend of Alice, and Bob subscribes to an application, then the application can access not only the public information of Alice, but also the private information of Alice so long as such a permission is granted by Bob. In the latter case, the application can access whatever Bob can access in Alice's

profile. Thus, while Alice does not subscribe to the application, she becomes a victim of an inference attack by virtue of her friend's subscribing to malicious applications.

3) Reflected: Rather than inferring information regarding the data owner (i.e., the subscriber or her friends), the attacker infers information about an individual referenced in the data. Consider a Reflected variant of Self Disclosure. The subscriber of an application writes on her own wall: "My wife was born in Vancouver." Now the application is aware of the birth place of the subscriber's wife. Consider a Reflected variant of Metadata Association. Suppose Bob subscribes to an application. He sends a message to his mother Alice, and explicitly address her as "Mom" in the message. The platform injects author and recipient IDs into the message entry. Now the application knows that Bob is a son of Alice, even if the friend list of Alice is private. In short, activities of a user may act as a mirror, allowing a malicious application to glance at the "reflection" of the victim in that mirror.

C. Discussion

An important cause of SNS API inference attacks is a mismatch between the user's privacy expectations and the privacy implications of adopting a certain permission configuration. By explicitly granting the `user_photos` permission (Example 2), one expects that photo accesses will be the only consequence. Users fail to incorporate into their mental model the complex inference closure entailed by releasing photo data. One solution is to reduce the information that can be inferred from the data released via the API. A second solution is to help refine user expectations, so that they are more aware of the privacy implications when they grant permissions to applications.

V. Risk Assessment

We propose a means to bridge the gap between the privacy expectation of the user, and the actual consequence of adopting a particular privacy setting. Specifically, we propose a risk assessment scheme (or a ***risk metric***) that provides users with a scalar value measuring how easy it is for a malicious application to launch an inference attack. The metric uses the authentication attack described in Section III as a proxy for SNS API inference attacks, and measures the probability that the authentication attack can be launched against the subscriber of a third-party application.

A. A Risk Metric

The object of measurement is a specific configuration of privacy settings that the user adopts for a given application. Such a configuration includes permissions granted to the application, as well as other global privacy settings. We measure the risk induced by such a configuration. Let Γ be the space of all possible configurations of privacy settings for that user, and γ a typical member of Γ. We assume that Γ is partially ordered by $\sqsubseteq$, and we write $\gamma_1 \sqsubseteq \gamma_2$ iff γ_1 is no more permissive than γ_2.

We postulate that we can identify a set $\mathcal{Q} = \{q_0, q_1, \ldots, q_{N-1}\}$ of N authentication questions that are *typically* used by web applications for authentication purposes. (It is not necessary for $\mathcal{Q}$ to exhaustively cover all possible authentication questions, which is clearly impossible, but it should cover a small but representative set.) Associated with the questions is a set $\mathcal{A} = \{a_0, a_1, \ldots, a_{N-1}\}$ of canonical inference algorithms, one for inferring the answer of each question. $\mathcal{Q}$ and $\mathcal{A}$ together forms our benchmark. Suppose further that a typical web application will ask users to answer k of the questions, selected randomly from $\mathcal{Q}$ (with equal probability). Our risk metric measures the probability $\pi_k(\gamma)$ that an attacker can use the inference algorithms in $\mathcal{A}$ to successfully infer the correct answers to k randomly selected questions.

For the metric $\pi_k(\gamma)$ to be defined properly, we need a number of parameters for the model.

- $F_i : \Gamma \to \{0, 1\}$ is a ***feasibility predicate*** indicating whether the data needed by a_i to attempt inferring an answer to question q_i is available when the user adopts a given configuration $\gamma \in \Gamma$ of privacy settings. The predicate F_i must be monotonic with respect to $\sqsubseteq$. That is, $\gamma_1 \sqsubseteq \gamma_2$ implies $F_i(\gamma_1) \Rightarrow F_i(\gamma_2)$.
- $P_i \in [0, 1]$ is a conditional probability, called ***inference accuracy***. Given that the privacy settings of the user allows the attacker to access the data needed for inference, P_i is the probability that the attacker can correctly infer the answer to question q_i. This probability is induced by the distribution of profile data among the users of the SNS. We further assume that the successful inference of each question is independent from the inference of other questions. Further justification of this assumption will be given in the sequel.

We can compute $\pi_k(\gamma)$ as follows:

$$\sum_{S \in [\mathcal{Z}_N]^k} \frac{\prod_{i \in S} F_i(\gamma) \times P_i}{\binom{N}{k}}$$

where $\mathcal{Z}_N = \{0, 1, \ldots, N-1\}$, and $[\mathcal{S}]^k$ is the set of all size-k subsets of set $\mathcal{S}$. If F_i and P_i are available, then it takes $O(N^k)$ time to compute $\pi_k(\gamma)$.

B. Model Parameters

To deploy the risk metric, one has to select the benchmark $\mathcal{Q}$ and $\mathcal{A}$, and supply the feasibility predicate F_i as well as the inference accuracy P_i for each inference algorithm $a_i \in \mathcal{A}$. We discuss guidelines on how this can be achieved.

The questions in $\mathcal{Q}$ should be selected in such a way that covers a diverse range of data sources described in Section IV-A. In order for the assumption of independence among P_i to be realistic, the inference algorithms in $\mathcal{A}$ shall draw on relatively independent data sources. The two inference algorithms in Examples 1 and 2 are good examples: the first one works with wall messages, while the second examines photo albums. Again, there is no need for $\mathcal{Q}$ to be exhaustive in its coverage of authentication questions. What is required

is that it contains a manageable but representative set of authentication questions.

Once q_i and a_i are fixed, the feasibility predicate F_i can be formulated by examining the Facebook API documentation, and enumerating the permissions required by a_i to access the relevant data sources.

The inference accuracy P_i of a_i must be assessed empirically. We suggest using real user data for this purpose. With consent from real users, we can download the profile data needed for making inference using dedicated third-party applications. Then we attempt to use inference algorithm a_i to infer an answer for authentication question q_i. The success rate of a_i gives us an estimation of P_i. We are currently seeking ethics approval for conducting such an estimation.

C. Usage

The risk metric can be deployed to help users to better assess the risk of granting permissions to applications. When a user subscribes to an application, or when the user adjusts the permissions granted to the application, the SNS can display the risk metric $\pi_k(\gamma)$ for, say, $k = 1, \ldots, 4$, where γ represents the current privacy settings γ with respect to that application.

VI. Conclusion and Future Work

In this work we identify the threat of SNS API inference attacks, provide a taxonomy of these attacks, and propose a risk assessment scheme to help users understand the risk of subscribing to a third-party application in an extensible SNS. Future work includes the creation of a concrete benchmark for the risk metric, the the extension of the metric to account for uneven popularity of authentication questions, and the design of a secure API for extensible SNSs.

An ongoing work is to actually create a benchmark, formulate the feasibility predicates, and empirically assess the inference accuracy of the inference algorithms in the benchmark. This would allow us to empirically evaluate the effectiveness of the risk assessment scheme.

One limitation of the proposed risk assessment scheme is that it assumes all authentication questions in the benchmark are equally popular. An improvement is to reformulate the metric so that it takes into account the uneven popularity of the authentication questions. An interesting research question would be to determine which version of the risk metric is actually more effective in steering users' privacy expectations.

Bridging the privacy expectation of the user and the actual risk of subscribing to applications is one of the two solution approaches discussed in Section IV. The other approach is to redesign the platform API and the data structures of an extensible SNS to minimize inference. We are interested in pursuing this direction.

Acknowledgment

The authors would like to thank Jun Biao Zhang for helping to identify the permissions required by the example attacks, and Ida Siahaan and Philipp Woelfel for their helpful discussions. This work is partly funded by an NSERC Strategic Project Grant.

References

[1] E. Steel and G. A. Fowler, "Facebook in privacy breach," *The Wall Street Journal*, Oct. 2010.

[2] J. He, W. W. Chu, and Z. V. Liu, "Inferring privacy information from social network," in *Proceedings of ISI'06*, ser. LNCS, vol. 3975. San Diego, CA, USA: Springer, May 2006, pp. 154–165.

[3] W. Xu, X. Zhou, and L. Li, "Inferring privacy information via social relations," in *Proceedings of the 24th IEEE ICDE Workshop*, Cancun, Mexico, Apr. 2008.

[4] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All your contacts are belong to us," in *Proceedings of WWW'09*, Madrid, Spain, Apr. 2009, pp. 551–560.

[5] E. Zheleva and L. Getoor, "To join or not to join," in *Proc. WWW'09*, Madrid, Spain, Apr. 2009, pp. 531–540.

[6] J. Lindamood, R. Heatherly, B. Thuraisingham, and M. Kantarcioglu, "Inferring private information using social network data," in *Proceedings of the 18th International Conference on World Wide Web (WWW'09)*, Madrid, Spain, Apr. 2009, pp. 1145–1146.

[7] A. Besmer, H. R. Lipford, M. Shehab, and G. Cheek, "Social applications: Exploring a more secure framework," in *Proceedings of the 2009 Symposium On Usable Privacy and Security (SOUPS'09)*, Mountain View, CA, USA, Jul. 2009.

[8] A. Felt and D. Evans, "Privacy protection for social networking platforms," in *Web 2.0 Security and Privacy (W2SP'08)*, Oakland, CA, USA, May 2008.

[9] D. J. Solove, *The Digital Person: Technology and Privacy in the Information Age*. NYU Press, 2004.