

## A Practical Attack to De-Anonymize Social Network Users

Gilbert Wondracek    Thorsten Holz  
*Technical University Vienna,  
 Austria*

Engin Kirda  
*Institute Eurecom,  
 Sophia Antipolis*

Christopher Kruegel  
*University of California,  
 Santa Barbara*

**Abstract**—Social networking sites such as Facebook, LinkedIn, and Xing have been reporting exponential growth rates. These sites have millions of registered users, and they are interesting from a security and privacy point of view because they store large amounts of sensitive personal user data.

In this paper, we introduce a novel de-anonymization attack that exploits group membership information that is available on social networking sites. More precisely, we show that information about the group memberships of a user (i.e., the groups of a social network to which a user belongs) is often sufficient to uniquely identify this user, or, at least, to significantly reduce the set of possible candidates. To determine the group membership of a user, we leverage well-known web browser history stealing attacks. Thus, whenever a social network user visits a malicious website, this website can launch our de-anonymization attack and learn the identity of its visitors.

The implications of our attack are manifold, since it requires a low effort and has the potential to affect millions of social networking users. We perform both a theoretical analysis and empirical measurements to demonstrate the feasibility of our attack against Xing, a medium-sized social network with more than eight million members that is mainly used for business relationships. Our analysis suggests that about 42% of the users that use groups can be uniquely identified, while for 90%, we can reduce the candidate set to less than 2,912 persons. Furthermore, we explored other, larger social networks and performed experiments that suggest that users of Facebook and LinkedIn are equally vulnerable (although attacks would require more resources on the side of the attacker). An analysis of an additional five social networks indicates that they are also prone to our attack.

### I. INTRODUCTION

*Social networking sites* such as Facebook, LinkedIn, Twitter, and Xing have been increasingly gaining in popularity [1]. In fact, Facebook has been reporting growth rates as high as 3% per week, with more than 300 million registered users as of September 2009 [2]. Furthermore, this site has more than 30 billion page views per month, and it is reported to be the largest photo storage site on the web with over one billion uploaded photos. Clearly, popular social networking sites are critical with respect to security and especially privacy due to their very large user base.

Of course, social networking sites are not less secure than other types of websites. However, the difference to other sites lies in the amount of private and possibly sensitive data that they store. Social networking sites are typically used to contact friends, discuss specific topics in online discussion

forums and groups, establish new business contacts, or simply to keep in touch with other people. Along with the information about friendships and acquaintances, users often provide a great deal of personal information that may be interesting for attackers. Although social networking sites employ mechanisms to protect the privacy of their users, there is always the risk that an attacker can correlate data or abuse the structure of a social network to infer information about registered individuals [3]–[5].

In this paper, we introduce a novel de-anonymization attack against users of social networking sites. In particular, we show that information about the *group memberships* of a user (i.e., the groups of a social network to which a user belongs) is often sufficient to uniquely identify this user. When unique identification is not possible, then the attack might still significantly reduce the size of the set of candidates that the victim belongs to.

To make the de-anonymization attack practical, we present a way in which an adversary can learn information about the group memberships of a user who is just browsing the web. To do this, an attacker can leverage the well-known technique of *history stealing* [6], [7]. More precisely, using history stealing, an attacker can probe the browser history of a victim for certain URLs that reveal group memberships on a social network. By combining this information with previously collected group membership data from the social network, it is possible to de-anonymize any user (of this social network) who visits the attacker’s website. In some cases, this allows an attacker who operates a malicious website to uniquely identify his visitors by their name (or, more precisely, the names used on the corresponding social network profiles).

Previous work in the area of de-anonymization was mostly focusing on correlating information from several independent data sets (datasets from different sources). For example, Griffith and Jakobsson used public records such as marriage and birth information to derive a mother’s maiden name [8]. Narayanan and Shmatikov showed, in two recent papers, that information from different data sources can be combined to de-anonymize a user [4], [9]. In contrast, our attack uses only information from a single social networking site, and combines it with intrinsic information that is generated while users access the site. That is, our attack makes use of the fact that the browser records the URLs of the social networking

site that a user visits (since browsers typically keep an access history for some time).

To demonstrate that our attack works, we performed both a theoretical analysis and empirical measurements for users of the Xing social network. The results suggest that our attack can be used to potentially de-anonymize millions of users. Due to the limited resources that were available to us, we focused our empirical evaluation on Xing, a medium-sized network that has eight million registered users. We managed to extensively collect data for this network and achieved a high coverage of its groups and members. However, to demonstrate that the attack is not conceptually limited to one social network, we also performed an empirical feasibility study on two other, significantly larger networks: Facebook and LinkedIn. Furthermore, we also briefly studied five other social networks and found that they are also vulnerable to our attack.

Our attack can also be generalized to other websites that generate *sparse datasets* (i.e., the information about each individual user covers only a small fraction of the overall attributes) [9]. In the case of social networks, our attack works because even the most active user is only a member of a small fraction of all groups, and thus, the group membership information serves as a fingerprint. Sparse datasets are common with websites that deal with user data. For example, Amazon and eBay use concepts similar to groups on social networks (“Customer Communities” and “Groups,” respectively), meaning that they are both potentially vulnerable to our de-anonymization attack. In summary, we make the following three contributions in this paper:

- We introduce a novel de-anonymization attack, and show how it can be applied to social networking sites. The key idea of the attack is to exploit information about specific users (in this case, membership in groups) combined with the victim’s browsing history.
- We demonstrate several techniques to scale our attacks to real-world social networking sites. This is a challenging task since these websites often have tens of millions of users.
- We provide both a theoretical analysis and empirical measurements to demonstrate the feasibility of our attack. The results indicate that about 42% of users in the social network Xing can be reliably de-anonymized. Furthermore, we empirically show that both Facebook and LinkedIn are also potentially vulnerable.

## II. BACKGROUND

In this section, we provide a brief introduction to the background concepts to allow the reader to better understand our attack. We first present a model of social networks, and define the terminology we use within this paper. We then list our assumptions about the attacker. We continue with an overview of the common structure of social networks, and

discuss the aspects of this structure that we exploit. Finally, we explain why social networks are commonly prone to history stealing attacks.

### A. Model and Definitions

Throughout this paper, we use the following models and definitions to describe our de-anonymization attack.

*Social Networks:* A social network  $S$  is modeled as a graph  $G = (V, E)$  containing nodes  $V$  representing users, and undirected edges  $E$  representing the “friendship” relation between two users. Furthermore, the social network contains  $G$  groups. Each group  $g \in G$  contains a set of users from  $V$ :  $\forall g \in G : g \subseteq V$ . Social networks typically do not allow empty groups without any user (and also actively delete such groups). Thus, we can assume, without loss of generality, that  $\forall g \in G : g \neq \emptyset$ .

Each user  $v \in V$  is a member of  $n$  groups, with  $n \geq 0$ . We model this information as a vector  $\Gamma(v) := (\Gamma_g(v))_{g \in G}$  such that:

$$\Gamma_g(v) = \begin{cases} 1 & \text{if } v \text{ is a member of group } g \\ 0 & \text{if } v \text{ is not a member of group } g \end{cases} \quad (1)$$

For each group  $g$  in which  $v$  is a member, one dimension of  $\Gamma(v)$  is set to one. Otherwise, this dimension is set to zero. For the case of  $n = 0$  (i.e., the user is not a member in any group), the vector  $\Gamma(v)$  contains only zeros. This is the worst case for our attack.

As we will show, the vector  $\Gamma(v)$  can be used to de-anonymize users within a social network. In particular,  $\Gamma(v)$  serves as the *group fingerprint* of a user, and we demonstrate in our experiments that this fingerprint, in practice, is characteristic for a given user.

*Browser History:* A building block that we use during our attack is the browsing history  $\beta_v$  of a user  $v$ . A web browser maintains a list of web pages that the user has recently visited. Every time a user visits a page  $p$ , the URL  $\phi_p$  that was used to load this page is added to  $\beta_v$ . Moreover, entries in  $\beta_v$  expire. That is, after a time interval  $\tau$  has elapsed, the URL related to  $p$  is removed from  $\beta_v$ . The timeout itself depends on the browser and user settings. For example, Mozilla Firefox uses  $\tau = 90$  days by default, while Microsoft Internet Explorer only uses  $\tau = 20$  days. Apple Safari is between both browsers with  $\tau = 1$  month by default, whereas Google Chrome has an unlimited history timeout  $\tau = \infty$ .

*Attacker Model:* We have two basic assumptions about an attacker. First, we assume that the attacker can determine which web pages, from a given set, a specific user  $v$  has accessed in the recent past (within time  $\tau$ ). This means that the attacker can determine whether or not a given URL  $\phi_p$  is in  $\beta_v$ . The attacker, thus, has a method to compute, for a given victim  $v$ , the function  $\sigma_v(\phi_p)$ , which is defined as

follows:

$$\sigma_v(\phi_p) = \begin{cases} 1 & \text{if } \phi_p \in \beta_v \text{ for the user } v \\ 0 & \text{if } \phi_p \notin \beta_v \text{ for the user } v \end{cases} \quad (2)$$

It is reasonable to assume that such a function exists and that the attacker can perform the computation based on *history stealing*, as we show in Section II-C.

The second assumption is that the attacker has a way to learn about the members of groups for a given social network  $S$ . As defined above, a group  $g$  is a non-empty subset of the overall users of  $S$ . The attacker does not need to have the membership information for all groups  $g \in G$ . However, knowledge about more groups makes the attack more efficient. In Section III-C, we discuss how an attacker can obtain the necessary group membership information.

We believe that our two assumptions about an attacker can be (easily) satisfied in practice, and our empirical experiments support this claim. Moreover, as we will discuss in Section III, our attack is able to tolerate a certain amount of inaccuracy. That is, even when the history stealing attack does not produce a perfect group fingerprint  $\Gamma(v)$  for a victim  $v$ , or when the attacker’s view of the social network is different than the network’s actual state (e.g., due to users who join and leave groups), the attack can still be successful. However, in such cases, it typically proceeds slower and produces larger candidate sets.

### B. Structure of Social Networking Sites

1) *Overview*: Most social networking sites share the same basic structure. Each user  $v$  within the network has a *profile*  $p_v$  that contains (partially sensitive) information. This information, for example, can be the user’s full name, photographs, date of birth, relationship status, former and current employers, and education. One of the core technical components of a social network is its website, and the underlying web application. The web application provides the main functionalities of the social network. This functionality often comprises of features that allow a web visitor to become a member, to edit personal profiles, to view other user profiles, or to join groups. To become a member of a social network, users can sign up at the website. This process usually only requires a valid e-mail address for verification purposes.

Since social networks can have millions of users, most popular social networks (see Table I) include features that allow users to be organized in *groups*. This feature allows users of a social network to easily find other users with whom they might share specific interests (e.g., same hobbies), demographic groups (e.g., studying at the same university), political or sexual-orientation, and even medical conditions. Typically, there exists some kind of hierarchy within a group. That is, particular members can hold the role of administrators or moderators, which grants them some special privileges (e.g., sending messages to the whole

group, or removing members). In general, two different types of groups exist:

- *Public groups*: These groups allow all members of the social network to join. Typically, members are automatically added to the group when they wish to join. Interestingly, we found that some social networks even allow non-group members to list the members of public groups (e.g., Facebook).
- *Closed groups*: On the other hand, *closed groups* require some sort of authorization before a member is allowed to join. In practice, this means that a group administrator or moderator needs to manually approve each membership request.

The different social networks vary widely in the number of available groups. Networks that target a general audience typically have a large number of groups, and the average user is a member of many groups. Social networks that target business users, on the other hand, have a smaller number of groups, and the average user is only a member in a few groups (see Section V for more specific results).

2) *Web Applications*: The web applications for the most popular social networks (see Table I) rely on hyperlinks and HTTP GET parameters to implement the communication between a user (more precisely, her browser) and the actual web application. For example, Figure 1 shows four real-world examples from the web application of Facebook and Xing that are representative for two groups of hyperlinks. The first link is used to tell the web application to display the currently logged-in user’s “home” area. Since the hyperlink for this operation is the same for every user of the social network, we refer to links of this type as *static hyperlinks*. In contrast, the other links are used to inform the web application of state changes requested by a user. For example, the second link sends a request to the web application that the user with the ID `userID` wishes to upload a new profile picture. This link contains a dynamic token (in this case, the ID of user  $v$ ), so we consequently call it a *dynamic hyperlink*. This type of links explicitly contains information about a user since the link is unique for each user of the social network (i.e., this link identifies a particular  $v \in V$ ).

Besides dynamic hyperlinks that contain information about users, there also exist dynamic hyperlinks that contain (or embed) information about groups. The third and fourth link of Figure 1 show two examples in which information about a specific group (i.e., a `groupID` parameter) is encoded within the hyperlink. Note that each link uniquely refers a group  $g \in G$ .

From the web application’s point of view, these hyperlinks facilitate the “internal” state keeping and communication between the web application and the user’s web browser. Since web browsers are agnostic to the semantic interpretation of links, they simply add the URLs of all visited web pages to the browsing history  $\beta_v$  of a user  $v$ . Note that since the interesting information is already encoded in the URL itself,

Name of social network	# users	Focus	Alexa traffic rank [1]	Supports groups
Facebook	300,000,000+	general audience, worldwide	2	✓
MySpace	260,000,000+	music, worldwide	11	✓
Friendster	90,000,000+	general audience, worldwide	111	✓
LinkedIn	50,000,000+	business, worldwide	53	✓
StudiVZ	15,000,000+	students, Germany	179	✓
Xing	8,000,000+	business, Europe	285	✓
Biggadda	5,500,000+	teenage audience, India	3,082	✓
Kiwibox	2,500,000+	teenage audience, worldwide	74,568	✓

Table I: Overview of popular social networking websites. The data is based on information provided by individual social networks, public sources such as Alexa [1], and our analysis.

- (1) `http://www.facebook.com/home.php?ref=home+`
- (2) `http://www.facebook.com/ajax/profile/picture/upload.php?id=[userID]+`
- (3) `http://www.facebook.com/group.php?gid=[groupID]&v=info&ref=nf+`
- (4) `https://www.xing.com/net/[groupID]/forums+`

Figure 1: Examples of distinct types of web application hyperlinks for different social networks.

it does not matter if the website is using security-enhancing protocols such as HTTPS for protecting the actual content. The URL is nevertheless added to the browser’s history. From an attacker’s point of view, this behavior is interesting, since it enables the attacker to identify groups a user has visited, and even potentially identify a specific user. That is, if the attacker is able to determine which pages are in the victim’s browsing history (i.e., she can compute the function  $\sigma_v(\phi_p)$  for pages loaded via dynamic hyperlinks  $\phi_p$ ), she can use this information to de-anonymize a user  $v$  (as shown in more detail later).

### C. History Stealing

*History stealing* is a known attack in which a malicious website can extract the browsing history of a visitor. One of the first descriptions of this attack dates back to the year 2000 [10], and the technique was re-discovered several times in the recent years (e.g., [11], [12]). The core observation behind this attack is the fact that a web browser treats hyperlinks differently depending on whether or not a hyperlink was previously accessed by a user. This means that a browser implements the function  $\sigma_v(\phi_p)$  (that is, the browser implicitly checks whether a target URL  $\phi_p$  is in the browsing history  $\beta_v$ ). Typically, hyperlinks to web pages in the browsing history are displayed in a different color to indicate to the user that this link has been clicked in the past. An attacker can use various techniques to probe whether or not a web page is in the browsing history:

- An attacker can create an HTML page with links to target web pages of interest and use background image tags in the `a:visited` style information. Since images can be referenced with URLs, the user’s browser will then access these URLs if a target web page is in  $\beta_v$ .

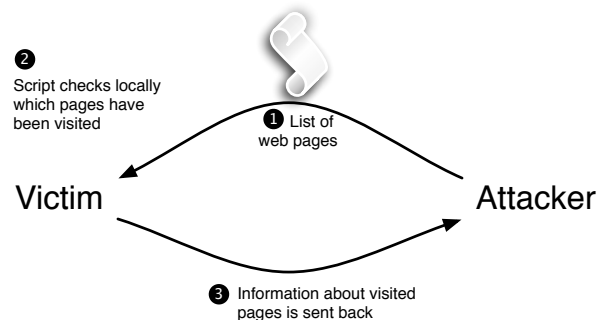


Figure 2: Schematic overview of history stealing attack.

- Alternatively, an attacker can also use client-side scripting (for example, JavaScript) to generate and iterate over a list of target links and programmatically check for the `a:visited` style to see if a link is present in  $\beta_v$ .

Note that an attacker has to probe for each URL (and cannot simply access the full browsing history of a victim), obtaining one bit of information per URL that determines whether it is contained in  $\beta_v$  or not.

From a schematic point of view, each attack scenario is the same (see Figure 2): First, the attacker sends a list of URLs to the victim’s browser. Second, the attacker forces the browser to check for each URL whether or not it is contained in the browsing history using one of the methods discussed above. Finally, a report is sent back to the attacker, who then obtains a list of URLs that are contained in the victim’s browsing history.

History stealing can be used for different kinds of attacks. For example, it can be used for more effective phishing. Jakobsson and Stamm presented a scenario where the attacker checks the browsing history for various bank site URLs. If the attacker sees that the victim has visited a certain

bank, she can then launch targeted phishing attacks [7] that target this bank. In the context of web applications, this means that an attacker can apply this technique to reconstruct knowledge on past interaction between the victim and the web application. While this knowledge alone might not be sufficient for many attack scenarios (e.g., an attacker would still need the online banking credentials to withdraw funds – requiring a phishing step), we show that we can successfully improve this technique to de-anonymize users of social networks.

All popular browsers (e.g., IE, Firefox, Safari) are vulnerable to history stealing attacks in their default settings (i.e., when the browser keeps a history of visited pages). To date, the problem has not been solved as it is often viewed as a usability feature/design issue rather than a browser bug.

#### D. Possible Attack Scenarios

De-anonymizing website visitors allows an adversary to launch targeted attacks against unsuspecting victims. Such attacks could be targeted phishing attempts [13], or could support social engineering efforts to spread malware (e.g., a message such as “Hello Mr. Gerald Stonart, we have discovered that your computer is infected. Please download and install this file.” might be displayed to Mr. Stonart). In addition, many people in political or corporate environments use social networks for professional communication (e.g., LinkedIn). Identifying these “high value” targets might be advantageous for the operator of a malicious website, revealing sensitive information about these individuals. For example, a politician or business operator might find it interesting to identify and de-anonymize any (business) competitors checking her website. Furthermore, our attack is a huge privacy breach: any website can determine the identity of a visitor, even if the victim uses techniques such as onion routing [14] to access the website – the browser nevertheless keeps the visited websites in the browsing history.

Of course, analogous to the situation where attackers compromise and abuse legitimate websites for drive-by downloads, the de-anonymization technique presented in this work can be used in a large-scale setup. That is, an attacker could abuse several compromised (but otherwise legitimate) websites as a vehicle for a de-anonymization attack.

### III. DE-ANONYMIZATION ATTACKS

With the background concepts introduced in the previous section, we now present our attack in more detail. We first introduce a basic variation of the attack, which is not feasible in practice. We then show how this basic approach can be refined to work for real-world social networks. Finally, we discuss how group membership information, a critical component for the advanced attack, can be obtained with a reasonable (limited) amount of resources.

#### A. Basic Attack

As mentioned in the previous section, certain dynamic hyperlinks contain explicit information about individual groups  $g \in G$  and users  $v \in V$  within a given social network  $S$ . An attacker can take advantage of this fact by using history stealing to probe for URLs that encode user information. In particular, the attacker can probe for a URL  $\phi$  that contains an identifier of user  $v$ . When a link is found that contains this identifier for  $v$ , then the attacker can reasonably assume that the browser was used by  $v$  in the past to access the user-specific URL  $\phi$ .

To find a suitable URL  $\phi$ , an attacker would first perform an information gathering step and join the target social network. In particular, he would analyze the website and look for dynamic hyperlinks that (a) contain identifiers that are indicative for a specific personal profile (e.g., because they contain a user ID) and (b) easy to predict for arbitrary users. For example, the second link in Figure 1 satisfies these properties: The user IDs are numerical and, hence, easy to predict. Also, the link is indicative for a specific user because the corresponding web application command (i.e., modifying the profile image) can only be performed by the owner of the profile. Thus, it is very unlikely that a user other than  $v$  has this link in her history.

Of course, this basic attack is not feasible in practice. The reason is that the attacker has to generate and check one URL for every user in the social network, and each potential victim’s browser would have to download all links and process them. In the case of Facebook, this would mean that more than 300 million links would have to be transferred to each victim. Thus, using the basic attack technique, the size of the search space (the candidate set) is far too large to be practical. In the following paragraphs, we show how group information can be used to significantly reduce the search space. Moreover, we need to keep in mind that the basic attack is still a valuable tool to identify a specific user among a (small) group of possible candidates.

#### B. Improved Attack

For our improved attack, we leverage the fact that many social network users are members in groups. Social networks commonly provide special features for groups in order to facilitate communication and interaction between group members. Often, discussion forums or mailing lists are provided. Since these features are incorporated into the social network’s web application, they are also prone to the history stealing technique. Similar to per-member actions, dynamic hyperlinks are used to incorporate group features into the web application. The main difference is that the identifiers in these links are not related to individual users within the group, but to the group itself. For example, the URLs (3) and (4) in Figure 1 are used for opening the group forums for two social networks.

An improved attack that leverages group membership information proceeds in two steps: First, the attacker needs to obtain group membership information from the social network. That is, the attacker has to learn, for some (possibly many) groups, who the members of these groups are. This step will be discussed in detail in the next section.

In the second step, the attacker uses history stealing to check the victim’s browser for URLs that indicate that this user has recently accessed a page related to group  $g$ , and hence, is likely a member of  $g$ . By preparing URLs for a set of  $n$  groups, the attacker can learn a partial group fingerprint of the victim  $\Gamma'(v)$ . More precisely, the attacker can learn the entry  $\Gamma_k(v)$  for each group  $k$  that is checked. The remaining entries are undefined. Clearly, being able to check more groups allows the attacker to learn more about the group fingerprint of a victim (i.e., he can obtain a larger, partial group fingerprint). This increases the chances that at least one entry of the partial group fingerprint is non-zero, which is necessary to be able to carry on with the attack.

Once the partial group fingerprint of a victim is obtained, the attacker checks for the presence of entries where  $\Gamma_k(v) = 1$ . Whenever such an entry is found, we assume that the victim  $v$  is member of the corresponding group  $k$ . At this point, the attack can continue in one of two ways.

A slower, but more robust, approach is to leverage the group membership information and generate a candidate set  $C$  that contains the *union* of all members  $\{u\}_k$  in those groups  $k$  for which  $\Gamma_k(v) = 1$ . That is,  $C = \cup\{u\}_k : \Gamma_k(v) = 1$ . Then, we use the basic attack for each element  $c$  in the candidate set  $C$ . More precisely, we use the basic attack to determine whether the victim  $v$  is one of the users  $c \in C$ . If so, then the attack was successful, and the user is successfully de-anonymized.

A faster, but more fragile, approach is to leverage the group membership information and generate a candidate set  $C$  that contains the *intersection* of all members  $\{u\}_k$  in those groups  $k$  for which  $\Gamma_k(v) = 1$ . That is,  $C = \cap\{u\}_k : \Gamma_k(v) = 1$ . Again, the basic attack is used to check for each user  $c$  in the candidate set  $C$ . Since the second technique uses set intersection instead of set union, it produces much smaller candidate sets and thus, it is faster.

**Robustness.** To see why the first attack is more robust than the second, we have to realize that the information that the attacker learns might be not entirely accurate. There are two reasons for this: First, the browsing history may contain incomplete information about the victim’s past group activity (e.g., a user might have deleted the browsing history at some point in the past). Second, the group membership information that the attacker has collected “degrades” over time, deviating increasingly from the real group and membership configuration as users join and leave groups.

As a result of inaccuracies, some entries  $\Gamma_k(v)$  in the partial group fingerprint might be wrong. Two cases need to be distinguished. The first case is that the entry  $\Gamma_k(v)$  for a

group  $k$  is 0 (or undefined), although  $v$  is a member of  $k$ . In general, this is not a problem, as long as the attacker finds at least one group  $k$  that the victim belongs to (and  $\Gamma_k(v) = 1$ ). The reason is the following. Since the entry for  $k$  is zero, the first attack will not add the members of  $k$  (including  $v$ ) to the candidate set  $C$ . However, we assume that there is another group that contains  $v$ . This means that  $v$  will be added to  $C$ , and the attack succeeds. For the second attack, the candidate set  $C$  can only shrink when a new group is considered (since set intersection is used). Thus, the attacker might need to check a larger candidate set, but he will still find  $v$  eventually.

The second case describes the situation where the entry  $\Gamma_k(v)$  for a group  $k$  is 1, although  $v$  is *not* a member of  $k$ . This causes no problem for the first attack, which simply adds additional users (all members from group  $k$ ) to the candidate set  $C$ . However, it is a problem for the second technique. The reason is that the intersection operation now includes a group that does not contain the victim user  $v$ . As a result,  $v$  will not be a part of the candidate set  $C$ , and hence, the attack will fail to find the victim.

In practice, an attacker would first attempt to use the fast (but fragile) approach based on set intersection. Only if this fails, one fall-backs onto the slower, more robust approach based on set union.

### C. Efficiently Obtaining Group Information

To carry out the advanced attack, the adversary requires information about groups and group memberships. In this section, we demonstrate how an attacker can obtain this knowledge with relatively little effort.

The number of groups is typically considerably smaller compared to the number of users. Nevertheless, collecting information about all groups and the members of each group is a challenging task. Therefore, we now discuss two techniques to efficiently obtain information about groups: *group directories* and *group member crawling*.

1) *Group Directory*: Typically, groups within social networks aim at attracting members that share a common interest with the group. To this end, social networks either offer a search functionality to find groups with a specific keyword, or they publish a list of the existing groups, called a *group directory*, via their website. This directory can be listed and searched by members of the social network to find groups related to their interests.

In our attack, it is desirable for the attacker to have knowledge on as many groups as possible. More specifically, the attacker is interested in the group identifiers to construct the hyperlinks for the history stealing attack. An attacker can use standard web crawling techniques to download the group directory, and then extract the group IDs from the web page’s source code. Several social networks even allow the group directory to be viewed by non-members, which

enables an attacker to use commercial crawling services for this task (see Section IV-C for details).

*Directory Reconstruction:* Some social networks do not publish a group directory or only do so partially (i.e., not all information about groups can be accessed this way). We implemented three methods to successfully circumvent this obstacle in practice.

First, the group identifiers that we observed in our experiments were either systematic (for example, numerical) or names. If group IDs can be guessed by an attacker, the group directory can be reconstructed by simply iterating over all (or at least a large fraction of) the ID space. The presence of the individual groups can be verified by trying to access each group’s web page. In Section V, we show that this brute-force technique can be used in practice effectively with a relatively small effort.

Second, an attacker can use the built-in search functionality of social networking websites to expose the group directory by listing all groups within a specific category of groups. Group search results are usually ranked by member size, which means that even if the result size is limited to a fixed value, an attacker gains valuable information.

Finally, we found that social networks may provide special “public” member profiles that can be accessed by non-members (i.e., they serve as “virtual” business cards). For privacy reasons, these profiles usually contain less personal information than the original member profiles. However, these public profiles typically reveal the groups for a specific member. In this case, an attacker can reconstruct the group directory (including the group members) by crawling the public member profiles. Note that this technique is rather costly, since it requires to crawl member profiles.

2) *Group Member Crawling:* In addition to group IDs, an attacker needs to obtain the IDs of the group members for a significant amount of groups to successfully de-anonymize group members. This step can also be automated and performed on a large-scale, as we discuss below.

If we deal with a public group, the easiest case is that the social network allows all members of this group to be listed. Then, we can use a standard crawling approach to discover the group members and use them for our attack. As we show in Section V, even tens of millions of group members can be crawled with only a limited amount of resources.

Some social networks limit the size of the member list that can be browsed. For example, Facebook only returns the first 6,000 members of a group. Hence, this limits a crawling attempt to only fully discover groups with up to 6,000 members. While this is still useful in practice, clearly, we would like to also be able to crawl larger groups.

In order to overcome this limitation, we take advantage of the fact that social networks typically allow searching within groups for members. This limits the amount of members returned per search, but we can efficiently extract most group members by searching for common first or last

names. We use publicly available data from the U.S. Census Bureau [15] to determine common names, and then utilize this information to search within large groups to extract their members.

If we are dealing with a closed group, we cannot easily access the membership information for this group since only members can access this information. Hence, we send a request to join the group from a legitimate user account by using a script (i.e., “I would like to become member of this group”). If our application is accepted, we leave the group after we have collected membership information. Surprisingly, such a simple automated demand is successful in practice as we show in Section V.

Note that, depending on the resources of an attacker, the member crawling may optionally be performed on-the-fly instead of offline before the actual attack. In an online setting, the attacker would crawl the groups a victim is a member of on demand, and then use the collected information for performing the second round of history stealing (i.e., verification step). From a conceptual point of view, both attacks are similar. They just vary in the amount of resources needed.

## IV. CRAWLING EXPERIMENTS

In this section, we describe our empirical experiments to extract group information from social networks and present the results we obtained for three social networks.

### A. Ethical and Legal Considerations

Crawling data in social networks is an ethically sensitive area. Clearly, one question that arises is if it is ethically acceptable and justifiable to conduct crawling experiments in social networks. Similar to the experiments conducted by Jakobsson et al. in [16], [17], we believe that realistic experiments are the only way to reliably estimate success rates of attacks in the real-world.

First, in the crawling experiments we conducted, we only accessed user information that was publicly available. Second, note that the crawler we wrote was not powerful enough to influence the performance of any social network we investigated. Third, the commercial crawling services we used had misuse protection mechanisms such as bandwidth throttling in place that prevented them from launching denial of service-like attacks against the websites that they were crawling (i.e., because of a higher crawling load).

We also consulted the legal department of our university (comparable to the IRB in the US), and we were informed that our experiments are approved.

### B. Overview

For our experiments, we performed an in-depth analysis of the Xing platform. Furthermore, we carried out feasibility studies for Facebook [2] and LinkedIn [18].

We chose these networks as they are representative of the different categories of popular social networks. For example,

Facebook aims at an audience that would like to maintain and create friendships, whereas LinkedIn and Xing are more focused towards business users who would like to maintain and extend their professional networks. Furthermore, each network has a unique way of representing its member and group directories.

Because of resource limitations, and because we had access to more Xing users for real-world user experiments, we chose to perform an in-depth analysis of Xing (Xing's size is considerably smaller than Facebook or LinkedIn, but it still has more than eight million users).

In the following, we discuss how an attacker can automatically extract group information (i.e., which is a prerequisite for the de-anonymization attack) from each of these social networks.

### C. Social Network Crawling Approaches

In order to access group information, an attacker can either run crawlers on her machines, or use third-party crawling services. For our experimental evaluation, we followed both approaches by implementing a custom web crawler, and by using commercial crawling services.

1) *Custom Crawler*: We implemented a web crawler that works by following the hyperlinks on a given starting public web page and then continues to download the HTML source code of each hyperlinked web page. To be able to also access parts of the social network that are only restricted to members, we added features that allow the crawler to login using provided member credentials. To this end, we manually registered three members to the social network using valid registration data (e.g., e-mail for our lab, etc.).

To extract the desired data, the crawler matches a set of regular expressions against the HTML source code. The extracted data (for example, group IDs and group names) are then stored in a database. To speed up the crawling process, we ran up to four instances of our crawler simultaneously.

*Anti-Crawling Techniques*: Most social networks employ anti-crawling techniques to protect the data of their members. Typically, if a member behaves suspiciously (for example, if he tries to access an overly large amount of user profiles in a short amount of time), this member's account will be temporarily, or permanently disabled. In contrast, no similar restrictions are in place for group directories. We believe that this mainly has two reasons:

- 1) The content is regarded as being public, and not security-relevant.
- 2) As a means of promoting groups, it should be intentionally easy to access the directory.

In addition, we observed that group directories often contain additional information that is relevant for an attacker (e.g., group size, or additional meta data). In our scenario, an attacker benefits from these factors, as it allows her to prepare the history stealing attack with relatively little effort.

2) *Commercial Crawling Services*: Attackers with limited computing or network resources might resort to commercial crawling services instead of operating their own web crawler. Such services allow customers to specify which websites to visit. Typically, the crawling service might accept a given list of web pages and regular expressions. Such a service can be very cost effective. For example, services such as 80legs [19] charge as low as \$0.25 per one million crawled URLs. In our experiments, we used such a service provider to perform some of our experiments.

### D. Crawling Experiments

We applied the two different crawling strategies to the three social networks Xing, Facebook and LinkedIn. In the following, we elaborate on how the group directories for each network can be retrieved, and provide an overview of the results.

1) *Xing*: We mainly concentrated our crawling experiments on this network, as its smaller size allowed us to fully crawl its public groups. This network is being actively policed by administrators who, for example, quickly remove empty, or inactive groups.

By directing our custom crawler to Xing, we could download the data of 6,574 groups containing more than 1.8 million unique members. Xing claims to have about 8 million members in total (i.e., including members that do not use groups at all). Hence, the users in these groups represent a substantial fraction of the entire social network.

*Closed Groups*: On Xing, the largest groups are *public*. That is, there are no restrictions on who is allowed to join these groups. On the other hand, an attacker might also be interested in crawling closed groups (that require manual approval by a moderator) in order to increase the effectiveness of the de-anonymization attack. To test how restrictive these groups are, we sent automated member requests from a legitimate account to the groups that had a large number of members.

We automatically applied for membership in 1,306 groups, and were accepted to 108 groups (8.2%). This allowed us, as an attacker, to see the user IDs of a total of 404,331 group members. Note that membership was denied by 1,199 groups (91.8%). However, despite the high rejection rate, we believe that our test demonstrates that a malicious user can successfully launch automated social engineering attacks to become member of closed groups. In practice, a real attacker would probably use fake fotos, detailed fake information, and a corresponding application text to increase her success rate. In our experiments, we simply asked if we could become member of the group.

Interestingly, our success rate was higher for the larger (i.e., more important from the attacker's point of view) groups. We were often instantly added to the group without receiving any feedback. Hence, membership application seems to be a formality for many large, closed groups.

2) *Facebook*: Recovering the group directory for Facebook initially appeared straightforward. The network publishes a group directory on its website. Access to the directory is not restricted. As a result, everyone can download it from the web. The directory itself is organized in a multi-level hierarchical collection of alphabetically ordered lists that provide pointers to individual web pages to make it convenient for a human to navigate.

Due to the large size of the dictionary, we decided to use a commercial crawling service to download it. In total, the dictionary consisted of 7.1GB of HTML data in about 7.4 million files that contain 39,156,580 group IDs. The crawling service cost us \$18.47 and we received the data after five days.

To enumerate Facebook’s group members, we extracted the group IDs from the group directory, and then used our custom crawler to enumerate the members for each group. Facebook permits each logged-in user to search within the member lists of arbitrary groups. This search can be used to filter the member lists to only show members whose first and/or last name fully matches the search token. Using an empty string as the search token returns random sample of the group members. The size of each search result is limited to 6,000 members, and can be retrieved in chunks of 100 members per HTTP request.

*Using Member Search Functionalities*: Since most Facebook groups have less than 6,000 members, this threshold is high enough and often allows us to obtain full member lists. An attacker can additionally use the search functionality to enumerate members in larger groups by searching for common first or last names. For example, with only 757 first names (233 male, 524 female), an attacker would be able to cover a cumulative percentage of more than 75% for each gender. According to the public 1990 US census [15] statistics, the most common first name, “James”, has a 3.318% probability among males, and 1.635% among the overall population. Hence, for groups with about 367,000 members, an attacker could obtain all members with this name (i.e., the search returns about 6,000 members for each name) on average. An attacker could even refine this approach by computing a more accurate name distribution by downloading Facebook’s own (also publicly available) member directory.

Note that enumerating very large groups that contain millions of members only provides a limited benefit for an attacker. Apart from the high crawling effort, the time required to check for so many members via history stealing would defeat the purpose of using groups as a means of search space reduction in a realistic attack scenario (e.g., see throughput rates in Section V-C). However, an attacker can also use the group member search functionality to verify the membership of individual members.

*Results*: In total, we successfully crawled more than 43.2 million group members from 31,853 groups in a period

of 23 days using only two machines. While this is still only a fraction of the overall Facebook groups and members, it demonstrates the feasibility of the approach.

In general, we believe that an attacker could also use a malicious botnet in real-life, or crawl for a longer period of time, and collect significantly more data compared to our effort with only limited resources.

3) *LinkedIn*: Just like Xing, LinkedIn focuses on business users. LinkedIn is a popular service and is widely-known.

*Third-Party Crawling Use-Cases*: LinkedIn does not publish a full group directory, but provides a group search functionality for logged-in users. Theoretically, this functionality could be exploited in a similar fashion to the group member search functionality of Facebook. However, this requires a much larger effort due to the higher variation in possible group names as opposed to first or last names of individuals.

LinkedIn uses easy to predict group IDs. Specifically, LinkedIn uses numerical group IDs that range from 0 (oldest) to about 3,000,000 (the newest groups). In addition, the group ID space seems to be sparsely populated, as according to the network itself, it currently has only 442,434 registered groups.

In a two-phase crawling scenario, we first started a crawling pass with a commercial service [19]. In a preparation step, we first generated three million hyperlinks for the observed group ID space, and then used these links as “seed” for the commercial crawling service. The results of the crawling service can be used to identify which group IDs actually exist (i.e., a group profile page is returned if the ID exists). The cost for this experiment was \$7.49.

After probing for the existing groups, we performed a second crawling run to retrieve additional information for each group such as its group size and group description. As this operation requires a logged-in user, we had to use our custom crawler.

While LinkedIn restricts access to its group directory, we found this not to be the case for its public member directory. For privacy reasons, the public member profiles are much less detailed than the regular profiles within the social network. Surprisingly, though, they do contain the membership status and group IDs for all groups that a member has joined. Clearly, data on groups and group memberships does not seem to be regarded as being security-relevant. As the public member profiles can be freely accessed over the web, an attacker can use automated legal third-party services to fully “outsource” the information gathering phase of the de-anonymization attack.

In a different crawling scenario, we performed an experiment and used an external crawling service to crawl the public profiles of three million members that we randomly picked from LinkedIn’s member directory. The costs for the crawling were \$6.57. Assuming a linear cost model, we estimate overall costs of about \$88 for crawling all 40

	Facebook	MySpace	Friendster	LinkedIn	StudiVZ	Xing	Bigadda	Kiwibox
Uses dynamic links	✓	✓	✓	✓	✓	✓	✓	✓
Group directory	Full	Searchable	Full	Searchable	Searchable	Searchable	Searchable	Full
Member directory	Full	Searchable	Full	Full	Searchable	Searchable	Searchable	Searchable
Group member enumeration	≤6,000	Unlimited	Unlimited	≤500	Unlimited	Unlimited	Unlimited	Unlimited
Public member profiles	✓	✓	✓	✓	✓	✓	✓	×
Vulnerable	✓	✓	✓	✓	✓	✓	✓	✓

Table II: Vulnerability Comparison of Social Networks.

million public profiles. This small investment would allow an attacker to target all group members of LinkedIn in a de-anonymization attack.

4) *Other Social Networks*: In order to find out how generic the problem of group information disclosure is, we manually analyzed five additional popular social networks that share features with the three networks that we analyzed in more detail.

Table II shows the features that are related to our de-anonymization scenario for these networks. All networks are vulnerable to history stealing and de-anonymization via groups. While we did not conduct crawling experiments for these networks, we expect the results and techniques to be similar to the ones we described in this section. Our empirical results demonstrate that group memberships are generally not considered as being privacy-relevant in many social networks.

## V. EVALUATION

In this section, we evaluate the practical feasibility of the de-anonymization attack.

### A. Analytical Results

To assess the effectiveness of our de-anonymization attack, we first perform an analytical analysis of Xing. For this social network, we have a comprehensive overview. More precisely, we have crawled all public and several closed groups, together with all member information for these groups. We start with an overview of the different parameters of that network related to our attack scenario.

In total, we collected membership information for more than 1.8 million unique users in 6,466 public and 108 closed groups. Based on our data, the average Xing user is a member of 3.49 groups. By using data from Xing’s group directory, we found that for the whole network, the sum of group member sizes (i.e., the number of membership relations) for the 6,466 public groups totals to more than 5.7 million. In contrast, the 15,373 closed groups only contain about 4.4 million membership relations. Furthermore, the average size of public groups is 914 members, whereas it is only 296 members for closed groups. The closed groups that we crawled contain 404,548 unique users. However, of these users, 329,052 (81.34%) were already covered by the public groups in our dataset.

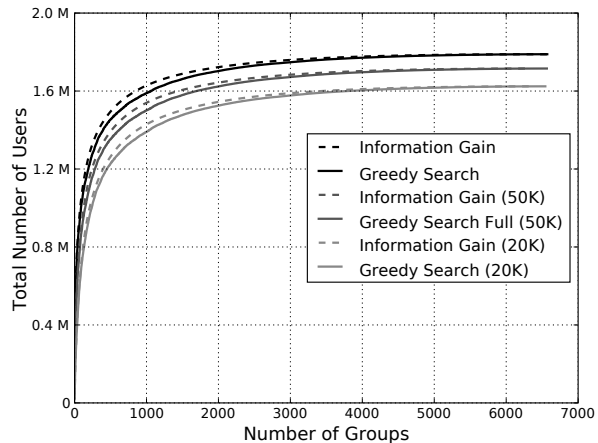


Figure 3: Cumulative distribution for number of unique users seen after crawling a specific number of groups (Xing).

These figures indicate, that an attacker can already collect a substantial amount of information by only focussing on public groups. The practical impact of closed groups appears to be rather low, given the increased effort that is necessary for gaining membership data for closed groups.

As mentioned in Section III, the improved de-anonymization attack requires that the history stealing step finds at least one group  $k$  that the victim is a member of (that is,  $\Gamma_k v = 1$  for at least one  $k$ ). Thus, when probing for an increasing number of groups, it is interesting to see how the number of unique users grows that appear in at least one of these groups. When this number grows quickly, the attacker has a good chance to get a “hit” after inspecting a small number of groups.

Of course, one also needs to consider the order in which the history stealing step should probe for group membership: Clearly, the attacker wants to optimize the process such that he sees each user at least once after as few attempts as possible. One approach is to perform a *greedy* search. That is, the attacker first probes the largest group, then the second largest group, and so on. The problem is that this order might not be optimal, since it does not take into account group membership overlaps. An alternative approach is to choose the groups by *information gain* (i.e., in each step, test the group which has most members *not* seen before). This might lead to a better probing procedure, since the

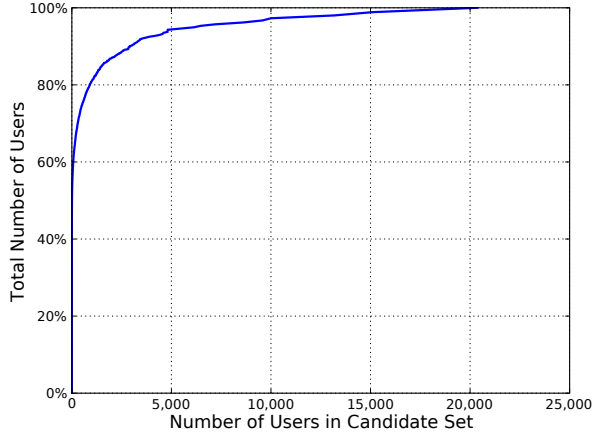


Figure 4: Cumulative distribution for the size of candidate sets (Xing).

attacker covers in each step the largest possible number of previously-unseen users.

Figure 3 shows the cumulative distribution for the number of unique users seen after crawling a specific number of groups. Besides considering all groups, we also studied the effect of limiting the search space to groups with at most 50,000 and 20,000 members, respectively. This takes into account that really large groups with hundreds of thousands of members are too large to probe efficiently. Thus, we restrict an attack to groups with an upper bound on the group size. The results indicate that an attack is very effective in practice: Even after testing only a few hundred groups, we have seen a significant percentage of all users at least once (i.e., a specific user is a member in at least one of the groups we have tested so far). In fact, we have seen more than 50% of the users after testing only 61 groups. After testing 1,108 groups, we have seen 90% of the users at least once. When restricting the search space, we can observe that we do not find each member at least once since some users are only member of large groups. Nevertheless, we can find more than 90% of the overall users when only considering groups smaller than 20,000 members.

Figure 3 also shows that the difference between the greedy and the information strategy is small. More precisely, the overall shape is very similar, only the number of tested group is significantly different: With the information gain strategy, an attacker only needs to probe 6,277 groups until he has seen each user at least once, whereas the brute-force approach requires to test 6,571 of all 6,574 groups before the complete set of users (who are part of at least one group) is covered.

For our next analysis, we assume that the attacker has successfully launched a history stealing attack and has computed an accurate group fingerprint  $\Gamma(v)$  for a victim  $v$ . This allows the attacker to use the fast de-anonymization attack based on set intersection. To demonstrate the effectiveness of

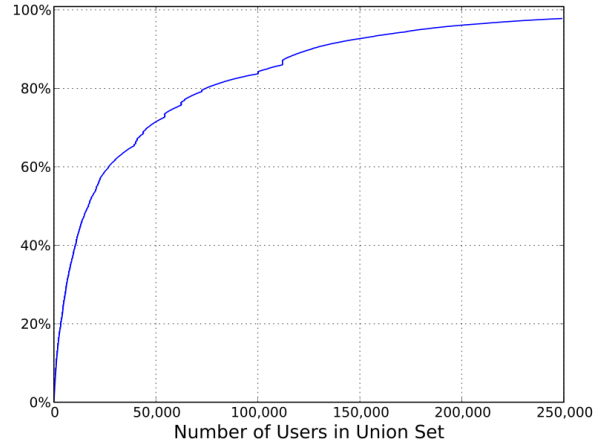


Figure 5: Cumulative distribution for the size of group union sets (Xing).

this attack, we show in Figure 4 the cumulative distribution of the candidate set sizes after set intersection. Each user in the candidate set needs to be inspected by the basic attack. Thus, a smaller size is favorable for the attacker. Interestingly, for 42.06% of the users (= 753,357 users), the group fingerprint is *exact*. That is, only a single user in the social network is a member of exactly these groups. These users can be uniquely identified just based on their group fingerprint, and no additional steps are required. For one million users, we can narrow down the candidate set to less than 32 users, and for 90% of all users, the candidate set is reduced from initially  $\sim 1.8$  million to less than 2,912 users. These results show that one can significantly narrow down the search space of candidates (who are then compared against the victim, one by one, using the basic attack).

Extracting a partial group fingerprint for a victim  $v$  might not always work flawlessly (for reasons discussed in Section III). As a result, the fast de-anonymization attack based on set intersection could fail. In this case, the attacker needs to resort to the slower but more robust attack based on set union. In Figure 5, we show the cumulative distribution of the candidate set sizes when performing set union. Compared to the candidate sets from Figure 4, the union sets are considerably larger due to the fact that we merge each group for which we have a match. Second, there is still a significant reduction in size compared to the overall number of users in Xing for a larger fraction of victims. For example, the set union attack still reduces the size of candidate set to less than 50,000 for more than 72% of all users on Xing.

We performed the same kind of analysis for Facebook. Since we did not completely crawl this social network, the results in Figure 6 provide an overview for the snapshot of group information we obtained during our experiment. For the 43.2 million users we have seen in 31,853 groups, 9.9 million have an exact group fingerprint  $\Gamma(v)$ . Furthermore,

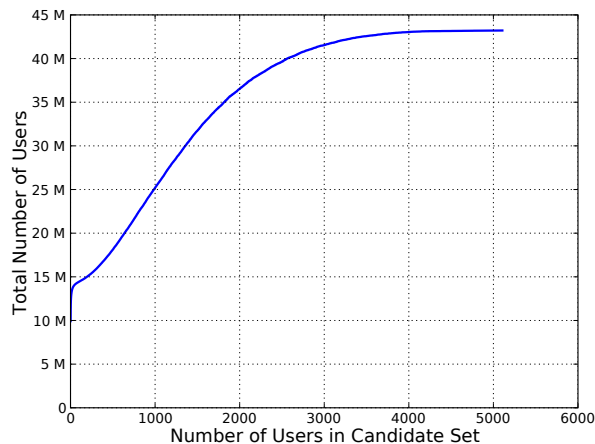


Figure 6: Cumulative distribution for the number of users seen in the candidate set (Facebook).

for 25.2 million users, the size of the candidate set is smaller than 1,000. The distribution of users in a candidate set of a given size is different compared to the Xing experiment, but we expect that the shape of the cumulative distribution to also be similar for Facebook if an attacker has crawled the complete network.

### B. Real-World Experiments

To demonstrate the practical feasibility of our attack, we created a website that performs a de-anonymization attack against Xing. While there is no technical restriction that limits our attack scheme to a specific network, we chose to only implement it for Xing, as our crawling experiments had covered the highest fraction of groups and members for this network.

For ethical reasons, we only promoted this website to volunteers that we picked from our personal contacts. Before the actual de-anonymization process starts, a warning page explains the purpose of the experiment to visitors. Then, the visitors need to explicitly acknowledge that they would like to participate in the experiment. Once a visitor decides to take part, she also needs to fill out a brief questionnaire that asks her about how she uses the social network, if she shares her computer or user account with someone else, and how often she participates in social networking groups.

Once the experiment starts, the visitor’s browsing history is probed for the URL of the main web page of Xing. If this URL is not present in the history, we assume that the visitor has not used Xing recently, or that she has disabled her browsing history (see the discussion in Section VI). In this case, we abort the de-anonymization attempt, and do not proceed.

If we do identify that a user has visited Xing before, we then use history stealing to probe for the URLs of the groups within Xing. That is, for each group, we check the visitor’s browsing history for dynamic group links. We only expect

these links to be present in the browsing history of users who are indeed members of these groups. We then perform the analysis based on the obtained group fingerprint, and present the result to the user.

*Results:* In total, we launched our attack on 26 volunteers from our Xing contacts. For 11 of these visitors, we could not find any dynamic links that indicated interaction with groups in their browsing history. The reasons for this can be manifold, for example only seldom usage of groups or regularly flushing the browsing history.

For the remaining 15 visitors, we could determine group fingerprints, and successfully launched de-anonymization attacks. More precisely, we could leverage the faster group intersection variant of the attack, and could compute intersection sets for 11 visitors. The median size of these intersection sets was only 570 members, thus a search within this set can be performed easily. For 4 visitors, we had to fallback to the more robust, but slower union set-based variant of our attack. As expected, the union set is significantly larger compared to the intersection set: the median size was 30,013 members, which can nevertheless be probed during a history stealing attack.

In summary, our experiments with real users show that our attack works in practice. We managed to de-anonymize 15 of 26 users who participated in our experiment.

### C. Run-Time and Throughput Rate

The runtime of the attack significantly influences its success rate in practice. Recall that we perform an active attack, and probe the victim’s browsing history. Thus, the victim needs to interact with the attacker for a certain amount of time. There are many techniques to convince a victim to stay longer at the attacker’s website. These techniques range from benign attempts such as showing a video or music clip, to offensive techniques such as “hijacking” the user’s browser with the help of JavaScript to prevent her from leaving the site. In any case, from the point of view of the attacker, it is desirable that the attack takes as little time as possible.

We measured the typical time it takes to perform a history stealing attack in practice. We performed experiments with the four major web browsers (i.e., Internet Explorer, Firefox, Safari, Chrome) on different operating systems (i.e., Windows Vista, Ubuntu Linux, Mac OS X). The test machine was a MacBook Pro with a 2.8 GHz Intel Core 2 Duo processor and 4 GB RAM. We booted the operating system natively on the machine, and used no virtualization software to prevent side-effects. In each test case, we measured the time it takes to perform a history stealing attack by checking a specific number of URLs (i.e., we check if these URLs have been visited or not): That is, we started with 1,000 URLs, and we increased the number of URLs to be checked in steps of thousand until we reached 90,000. We performed each test ten times, and calculated the mean values. These values are shown in Figure 7. In order to increase readability,

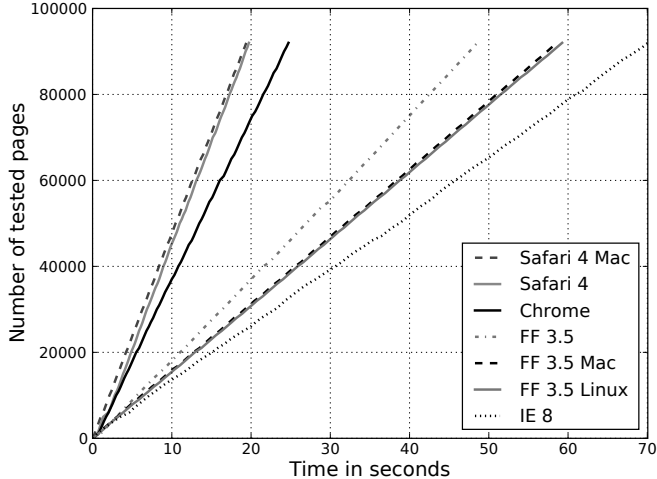


Figure 7: Runtime benchmark for different web browsers on different operating systems. The legend corresponds to the browsers from left to right.

and since the mean error between individual runs was always below 5%, we omit error bars.

Safari on both Mac OS X and Windows achieved the best results in our experiments: A history stealing attack with 90,000 tests can be performed in less than 20 seconds. Chrome is about 25% slower, while Firefox requires between 48 and 59 seconds, depending on the operating system. The slowest performance was measured for Internet Explorer, which took 70 seconds to probe all pages. Nevertheless, even for Internet Explorer, we could probe more than 13,000 URLs in less than 10 seconds. Together with the results from Figure 3, this show that an attacker can detect many groups of a victim in a small amount of time.

#### D. Fluctuation in Groups

Another aspect we need to consider is the fluctuation rate in groups. From the viewpoint of an attacker, it is interesting to understand how the groups and members in a social network change over time.

First, it is unlikely that an attacker has access to the networking and computing capacity that would enable her to take a complete snapshot of a social network (i.e., permit her to collect the necessary data for a de-anonymization attack in a time span that is short enough to prevent *any* change in both groups and members). In practice, depending on the size of the network and the attacker’s resources, the duration from start to end of crawling and the reconstruction of the groups directory might take days, or even weeks.

Second, there will also be changes that are caused by normal user behavior after the initial crawling phase. For example, members will join or leave groups, or new groups will be created. Over time, these changes cause the crawled data to increasingly deviate from the real configuration of groups and members in the network. Determining how stable

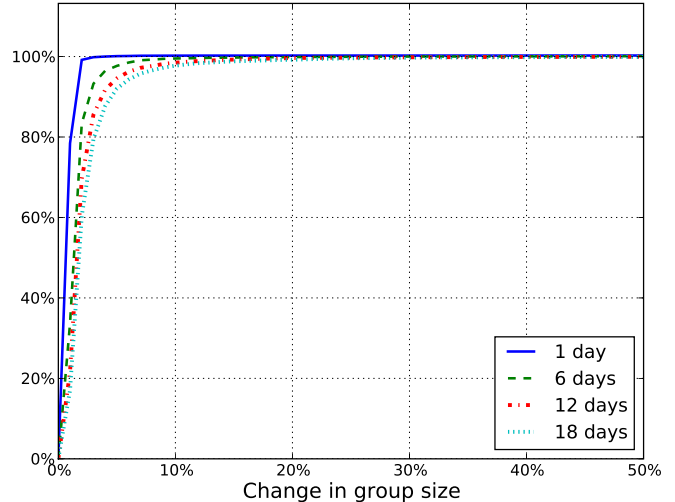


Figure 8: Degradation of group data in Xing.

the group data is, is related to the question of how often an attacker would have to recrawl parts or the entire social network. Hence, this directly influences how much effort an attacker has to invest for a de-anonymization attack. An attacker can also develop iterative approaches for keeping the collected information up-to-date, e.g. social networking features that explicitly allow the listing of only new members in groups and newly created groups can be used.

For measuring the fluctuation in groups, we conducted experiments for Xing. Instead of repeatedly crawling the entire network, we only downloaded the group directory and the member size for each group. This permitted us to repeat this operation every four hours over a period of 18 days.

Figure 8 shows the CDF for the changes in group size for four different periods. Interestingly, while the results from our measurements confirm that the quality of the collected group and member data degrades over time, they also show that the data stays relatively stable, significantly reducing the necessary crawling effort for an attacker.

While we are aware of the possibility that the amount of users that either join or leave a group might lead to the same overall group size, we believe that the result of our experiment is still accurate enough to give an indication of the amount of change that affects the Xing’s group configurations.

## VI. POSSIBLE MITIGATION TECHNIQUES

The approach presented in this work allow a malicious user to launch de-anonymization attacks against a large number of victims with relatively little effort. Whereas history stealing by itself is often not enough to identify individual users, combined with the misuse of group membership information stored in social networks, it becomes a critical weakness. In this section, we list mitigation techniques that aim to thwart our attack.

### A. Server-side Mitigation

As a server-side mitigation, web applications could use dynamic hyperlinks that an attacker cannot easily predict. For example, existing systems could be hardened against the history stealing attack by automatically adding HTTP GET parameters that contain random tokens to all hyperlinks. Depending on the utilized web server, it might be possible to retrofit existing web applications by using URL rewriting to automatically add such tokens to each URL.

Even adding a simple, alphanumeric string of length 2 would increase the attacker’s search space by a factor of 3844 ( $62^2$ ). Hence, the attack would effectively be prevented.

Also, web applications should preferably use HTTP POST instead of HTTP GET in order to send parameters. This is because only GET parameters are stored in the browsing history. In fact, a server-side mitigation solution that randomizes web-application links is presented in existing work [20].

Note that one difficulty with server-side mitigation is that the usability of the web applications may be affected. For example, it may become more difficult to bookmark parts of the application, or links to certain groups may become more difficult to remember.

### B. Client-side Mitigation

On the client-side, history stealing is more difficult to fix without sacrificing functionality. Obviously, the goal is to prevent browsers from leaking sensitive and private information via style information. As a solution, browsers could generally restrict client-side scripts from accessing the CSS properties of hyperlinks. Unfortunately, this could also break existing websites that legitimately do so.

In [6], the authors offer a clever solution by extending the *same-origin* concept of web browsers to visited links. Unfortunately, so far, none of the published countermeasures to history sniffing have experienced wide-spread adoption, whether on the server, nor on the client-side.

Current web browsers only provide limited options for protection against attacks that are based on history stealing. Because the attack can be implemented without the need for client-side scripting, turning off JavaScript, or using browser add-ons that protect against script-based attacks (for example, NoScript [21]) may only provide limited help.

Users can also permanently, or temporarily disable the browsing history. They can, for example, use the “private browsing modes” that are supported by several current browsers (e.g., Firefox, Safari). Unfortunately, all of these methods also require some effort on behalf of the user, and reduce the usability of web browsers and web applications.

## VII. RELATED WORK

Clearly, de-anonymization of privacy-sensitive data is not a new concept. Research initially focused on anonymization and de-anonymization of network level data. For example,

work by Pang et al. [22] presents techniques for anonymizing network packet traces with the intent of sharing data between researchers. As a reaction to anonymization research, Coulls et al. [23] introduced approaches that allow an attacker to de-anonymize network traces, and recover sensitive data on network topologies.

*Information Leakage and Social Networks:* Due to the popularity of social networks and the large amounts of sensitive data they store, the focus of de-anonymization research has recently extended to this area. Several publications have shown that seemingly non-sensitive data from publicly available sources can be used to recover private information about individuals. For example, Griffith and Jakobsson [8] use public records to infer individuals’ mothers’ maiden names, and Heatherly et al. [24], as well as Zheleva and Getoor [5], show how public data provided by social networks can be used to infer private information.

In addition, several publications have analyzed and measured features of social networks that are privacy-related. For example, Mislove et al. present a measurement study on social networks [25] while Bonneau and Preibusch evaluate the privacy settings and policies of a large number of social networks in [26]. Closely related to this context, several recent papers focus on scenarios for malicious activity directed against social networks. For example, Jagatic et al. evaluate the success rates of phishing attacks [13], and Brown et al. discuss context-aware spam [27]. Another study [28] by Bilge et al. shows the feasibility of automated identity theft attacks in social networks.

*Attacks on Browsing Privacy:* The de-anonymization scenario presented in this work leverages a browsing history stealing technique that is based on CSS and has been known since the year 2000. This technique has been discussed in several browser bug reports [10]–[12], and has been shown to be practical for targeted phishing attacks by Jakobsson and Stamm [7]. Despite its malicious potential, browser history stealing has not lead to any changes in browser software.

There are also other techniques that aim at exposing private browsing information. Several systems use timing properties to recover private information. For example, Felten and Schneider show an attack on web browsing history by analyzing caching operations [29], while Bortz and Boneh [30] use timing attacks to recover private information from web applications.

*De-Anonymization of Social Networks:* Narayan and Shmatikow have shown that statistical methods can be applied to de-anonymize micro-data by cross-correlating multiple datasets [9]. They extend their approach to social networks in [4], and prove that it is possible to de-anonymize members by mapping known, auxiliary information on the (social) network topology.

In [31], Diaz et al. present a de-anonymization approach that uses information gained from observing communication patterns between social network members.

In contrast to existing work, our attack uses only information from a single social networking site, and combines it with the browsing history of a user to identify individuals. Furthermore, our attack is highly practical, and works effectively in the real-world. In fact, as we demonstrate in the paper, the attack has the potential to affect the privacy of millions of registered social network users.

### VIII. CONCLUSION

In this paper, we introduce a novel, practical de-anonymization attack that makes use of the group information in social networking sites. Using empirical, real-world experiments, we show that the group membership of a user in a social network (i.e., the groups within a social network in which a user is a member), may reveal enough information about an individual user to identify her when visiting web pages from third parties.

The implications of the attack we present are manifold. The attack requires a low effort, and has the potential to affect millions of registered social networking users who have group memberships.

The theoretical analysis and empirical measurements we present demonstrate the feasibility of the attack on the Xing, Facebook, and LinkedIn social networks. Furthermore, our investigations suggest that many more social networks that support group memberships can potentially be misused for similar attacks.

### REFERENCES

- [1] Alexa, "Top 500 Global Sites," <http://www.alexa.com/topsites>, 2009.
- [2] "Facebook," <http://www.facebook.com>, 2009.
- [3] M. Chew, D. Balfanz, and B. Laurie, "(Under)mining Privacy in Social Networks," in *Proceedings of Web 2.0 Security and Privacy Workshop (W2SP)*, 2008.
- [4] A. Narayanan and V. Shmatikov, "De-anonymizing social networks," in *IEEE Symposium on Security and Privacy*, 2009.
- [5] E. Zheleva and L. Getoor, "To Join or Not To Join: The Illusion of Privacy in Social Networks with Mixed Public and Private User Profiles," in *18th International Conference on World Wide Web (WWW)*, 2009.
- [6] C. Jackson, A. Bortz, D. Boneh, and J. C. Mitchell, "Protecting Browser State From Web Privacy Attacks," in *15th International Conference on World Wide Web (WWW)*, 2006.
- [7] M. Jakobsson and S. Stamm, "Invasive Browser Sniffing and Countermeasures," in *15th International World Wide Web Conference*, 2006.
- [8] V. Griffith and M. Jakobsson, "Messin' with Texas, Deriving Mother's Maiden Names using Public Records," in *Third Conference on Applied Cryptography and Network Security (ACNS)*, June 2005.
- [9] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," in *IEEE Symposium on Security and Privacy*, 2008.
- [10] J. Ruderman, "CSS on a:visited can load an image and/or reveal if visitor been to a site," [https://bugzilla.mozilla.org/show\\_bug.cgi?id=57351](https://bugzilla.mozilla.org/show_bug.cgi?id=57351), 2000.
- [11] D. Baron, "a:visited support allows queries into global history," [https://bugzilla.mozilla.org/show\\_bug.cgi?id=147777](https://bugzilla.mozilla.org/show_bug.cgi?id=147777), 2002.
- [12] Z. Braniecki, "CSS allows to check history via :visited," [https://bugzilla.mozilla.org/show\\_bug.cgi?id=224954](https://bugzilla.mozilla.org/show_bug.cgi?id=224954), 2003.
- [13] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer, "Social phishing," *Commun. ACM*, vol. 50, no. 10, pp. 94–100, 2007.
- [14] R. Dingledine, N. Mathewson, and P. F. Syverson, "Tor: The Second-Generation Onion Router," in *USENIX Security Symposium*, 2004.
- [15] U.S. Census Bureau, "Frequently Occurring Names and Surnames," <http://www.census.gov/genealogy/www>, 2009.
- [16] M. Jakobsson, P. Finn, and N. Johnson, "Why and How to Perform Fraud Experiments," *Security & Privacy, IEEE*, vol. 6, no. 2, pp. 66–68, March-April 2008.
- [17] M. Jakobsson and J. Ratkiewicz, "Designing ethical phishing experiments: a study of (ROT13) rOnI query features," in *15th International Conference on World Wide Web (WWW)*, 2006.
- [18] "LinkedIn," <http://www.linkedin.com>, 2009.
- [19] Computational Crawling LP, "80legs," <http://www.80legs.com>, 2009.
- [20] M. Jakobsson and S. Stamm, "Web Camouflage: Protecting Your Clients from Browser-Sniffing Attacks," *IEEE Security and Privacy*, vol. 5, no. 6, 2007.
- [21] G. Maone, "NoScript," <https://addons.mozilla.org/de/firefox/addon/722>, 2009.
- [22] R. Pang, M. Allman, V. Paxson, and J. Lee, "The Devil and Packet Trace Anonymization," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, 2006.
- [23] S. Coulls, C. Wright, F. Monrose, M. Collins, and M. Reiter, "Playing Devil's Advocate: Inferring Sensitive Information from Anonymized Traces," in *Symposium on Network and Distributed Systems Security (NDSS)*, 2007.
- [24] R. Heatherly, M. Kantarcioglu, and B. Thuraisingham, "Preventing Private Information Inference Attacks on Social Networks," University of Texas at Dallas, Tech. Rep. UTDCS-03-09, 2009.
- [25] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and Analysis of Online Social Networks," in *7th ACM SIGCOMM Internet Measurement Conference (IMC)*, 2007.
- [26] J. Bonneau and S. Preibusch, "The Privacy Jungle: On the Market for Privacy in Social Networks," in *Eighth Workshop on the Economics of Information Security (WEIS)*, 2009.
- [27] G. Brown, T. Howe, M. Ihbe, A. Prakash, and K. Borders, "Social networks and context-aware spam," in *ACM Conference on Computer Supported Cooperative Work (CSCW)*, 2008.
- [28] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All Your Contacts Are Belong to Us: Automated Identity Theft Attacks on Social Networks," in *18th International Conference on World Wide Web (WWW)*, 2009.
- [29] E. W. Felten and M. A. Schneider, "Timing Attacks on Web Privacy," in *7th ACM Conference on Computer and Communications Security (CCS)*, 2000.
- [30] A. Bortz and D. Boneh, "Exposing Private Information by Timing Web Applications," in *16th International Conference on World Wide Web (WWW)*, 2007.
- [31] C. Diaz, C. Troncoso, and A. Serjantov, "On the Impact of Social Network Profiling on Anonymity," in *8th International Symposium on Privacy Enhancing Technologies (PETS)*, 2008.