All Your Contacts Are Belong to Us: Automated Identity Theft Attacks on Social Networks

Leyla Bilge, Thorsten Strufe, Davide Balzarotti, Engin Kirda
EURECOM
Sophia Antipolis, France
bilge@eurecom.fr, strufe@eurecom.fr, balzarotti@eurecom.fr, kirda@eurecom.fr

ABSTRACT

Social networking sites have been increasingly gaining popularity. Well-known sites such as Facebook have been reporting growth rates as high as 3% per week [5]. Many social networking sites have millions of registered users who use these sites to share photographs, contact long-lost friends, establish new business contacts and to keep in touch. In this paper, we investigate how easy it would be for a potential attacker to launch automated crawling and identity theft attacks against a number of popular social networking sites in order to gain access to a large volume of personal user information. The first attack we present is the automated identity theft of existing user profiles and sending of friend requests to the contacts of the cloned victim. The hope, from the attacker's point of view, is that the contacted users simply trust and accept the friend request. By establishing a friendship relationship with the contacts of a victim, the attacker is able to access the sensitive personal information provided by them. In the second, more advanced attack we present, we show that it is effective and feasible to launch an automated, cross-site profile cloning attack. In this attack, we are able to automatically create a forged profile in a network where the victim is not registered yet and contact the victim's friends who are registered on both networks. Our experimental results with real users show that the automated attacks we present are effective and feasible in practice.

Categories and Subject Descriptors

D.2.0 [**Software**]: Software Engineering : General; H.M [**Information Systems**]: Miscellaneous

General Terms

Security

Keywords

Social Network Security, Identity Theft

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

1. INTRODUCTION

A social network is a social structure that is made up of nodes representing individuals or organizations. These nodes may be tied to each other by properties such as friendship, common values, visions, ideas, business relationships and general interests. Although the idea of social networks has been around for a long time (e.g., see [14]), social networking web sites and services are a relatively new phenomenon on the Internet. Business relationship-focused social networking sites such as XING [13] (previously known as OpenBC) and LinkedIn [6], as well as friendship-focused social networking sites such as Facebook [4], MySpace [8], StudiVZ [11] and MeinVZ [7] have been gaining popularity among Internet users. In fact, LinkedIn boasts on its web site that it has 30 million registered users. XING, a business networking site that is very popular in Switzerland, Germany and Austria, claims to have 6 million registered users. Although it has only been created four years ago, Facebook now has more than 150 million active users and is reporting growth rates of 3% per week. According to Facebook, it registers 30 billion page views per month and is the largest photo storage site on the web with over 1 billion uploaded photos [5].

Unfortunately, as the interest for a new technology grows on the Internet, miscreants are attracted as well. For example, spam was not a major problem until the end of the '90s. However, as more and more people started using e-mail, unsolicited (i.e, spam) e-mails started increasing in numbers. In fact, spam has reached such high proportions that the Spamhouse Project [12] now estimates that about 90% of the incoming e-mail traffic in North America, Europa and Australasia is spam. Also, the increase in the popularity of e-mail also resulted in an increase in the numbers of malicious e-mails (e.g., e-mails with worm attachments, phishing e-mails, scam e-mails, etc.). Today, e-mail is a popular way of spreading infections.

As the popularity of social networking sites increase, so does their attractiveness for criminals. For example, worms have recently emerged that specifically target MySpace and Facebook users [9]. These worms make use of old ideas that are applied to a new technology. Analogous to classic worms such as LoveLetter [3] that used the contacts in a victim's Outlook address book to spread, these new social networking worms use the friend lists of a victim to send a copy of themselves to other social networking users. Although such e-mail attachments may raise more suspicion now as such tricks have already been seen by many e-mail users, they are not as well-known on social networking sites. Fur-

thermore, note that incoming e-mails with attachments are often scanned for malicious content and Bayesian filters are applied to sort out unsolicited mails. In comparison, social networking sites do not usually provide filtering mechanisms or warnings for dangerous content, hence, making it easier, in principle, for a potential attacker to send malicious applications and URLs to victims.

Fortunately, so far, social networking sites and services have been spared from large-scale, high profile attacks. Nevertheless, social networking sites are an attractive target for attackers because of the nature of the sensitive information that they contain on registered users. Typically, users enter their real e-mail addresses and provide information on their education, friends, professional background, activities they are involved in, their current relationship status and sometimes even list their previous relationships (e.g., on Facebook, one may read that Mr X. was together with Ms Y until they broke up in 2006). Hence, from the attacker's point of view, access to this type of detailed, personal information would be ideal for launching targeted, social engineering attacks, now often referred to as spear phishing [2, 1]. Furthermore, the collected e-mail addresses and personal information would be invaluable for spammers as they would 1) have access to e-mail addresses that belong to real people (i.e., one problem spammers face is that they often do not know if the e-mail addresses that they collect are indeed being used by real people or they are just secondary addresses that are not regularly read) and 2) have information about the people using these e-mail addresses allowing them to efficiently personalize their marketing activities, tailored according to the knowledge from the target's profile. Also, note that the ability to associate personal information with an e-mail address is important to be able to successfully bypass spam filters [21]. Such filters usually generate a list of "spammy" tokens versus "good" tokens after training with a large set of previously received e-mails. As a result, e-mails that contain the name of the user receiving the e-mail, or names of people that he is acquainted with tend to receive lower spam ratings than e-mails that are less personal. As a result, if the spammer is able to include some personal information in the spam that he is sending, he would be able to improve his chances of reaching the targeted user.

Typically, a prerequisite for being able to access personal information in a social networking site is to have a confirmed personal "relationship" with the person who is concerned. The default setting in Facebook is to allow all confirmed friends to have access to the personal information (e-mail address, photographs, etc.), but not to provide it to unconfirmed third parties. In LinkedIn, the contacts of a person can only be accessed if it is a confirmed business contact, and therefore he/she has already accepted a request and confirmed the relationship.

Hamiel and Moyer conducted an impersonation experiment in which they created a fake profile on LinkedIn for the well-known security expert Marcus Ranum. The authors obtained the information to create a plausible profile by manually surfing the web, visiting Ranum's personal web page, and his entry in Wikipedia [25]. By impersonating a high-profile person, the authors showed how effective an impersonization attack can be. The forged profile received many friend requests, even from one of the target's immediate family members.

In this paper, we investigate how easy it would be for a potential attacker to launch this type of impersonization attacks in an automated fashion against a number of popular social networking sites in order to gain access to a large volume of personal user information. Unlike a Sybil attack [17] where the attacker aims to subvert a reputation system of a peer to peer or a social network by creating a large number of pseudonymous entities, the attacks presented in this paper consist of the automated identity theft of real user profiles. In the first attack we clone an already existing profile in a social network and we send friend requests to the contacts of the victim. Hence, we are able to "steal" the contacts of a user by forging his identity and creating a second, identical profile in the same social network. Having access to the contacts of a victim, therefore, means that we can access the sensitive personal information provided by these contacts. Our experimental results show that a typical user tends to accept a friend request from a forged identity who is actually already a confirmed contact in their friend list.

In the second attack we present, we show that it is effective and feasible to launch an automated, *cross-site profile cloning* attack. In this attack, we are able to automatically identify users who are registered in one social network, but who are not registered in another. We can then clone the identity of a victim in the site where he is registered, and forge it in a social networking site where he is not registered yet. After we have successfully created the forged identity, we can then automatically attempt to rebuild the social network of the victim by contacting his friends that we have identified to be registered on both social networking sites. Our experimental results suggest that this attack is especially effective because profiles in this case only exist once on the social networking site that is being targeted. As a result, the friend requests that we send look perfectly legitimate and do not raise suspicion with the users who have been contacted.

We implemented our attacks in a prototype system called iCloner (identity Cloner). iCloner consists of several components that are able to crawl popular social networking sites, collect information on users, automatically create profiles, send friend requests and personal messages. Furthermore, iCloner also supports CAPTCHA [16] analysis and breaking capabilities that make our attacks feasible against social networking sites that employ CAPTCHAs to prevent automated access.

It is important to note that the attacks we present in this paper can potentially be launched on a large scale, allowing an attacker to control hundred of thousands of cloned accounts and thus reaching millions of real user profiles. Furthermore, if the attacker has a high number of different IP addresses at his disposal (such as a botnet that consists of thousands of compromised hosts), the detection of an automated attack like the ones presented in this paper may become more difficult.

The contributions of this paper are the following:

- We show that it is feasible in practice to launch automated attacks against five popular social networking sites. In particular, we present two automated identity theft attacks: Profile cloning and cross-site profile cloning.

- Even though some of the sites employ CAPTCHAs to prevent automated access, in some cases, there is

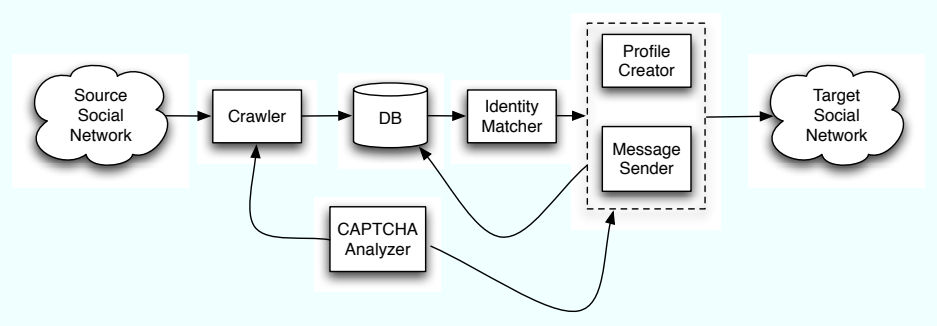

Figure 1: An architectural overview of iCloner

significant room for improvement to make these CAPTCHAs more difficult to break.

- We present experimental results with real users and show that the attacks we present are feasible in practice. Our results confirm empirically, as one would expect, that most social network users are not cautious when accepting friend requests or clicking on links that are sent to them. In fact, many are even willing to accept friend requests from people they do not know.

- We make suggestions on how social networking sites can improve their security, and therefore, better protect the privacy of their users.

The rest of the paper is structured as follows. In Section 2, we give a brief overview of iCloner, our prototype attack system. In Section 3, we describe the two cloning attacks that we used in our experiments to gain access to the victims' contacts. In Section 4, we discuss the results of our experiments. In Section 5, we discuss how social networking sites can improve their security. In Section 6, we discuss related work and conclude the paper in Section 7.

2. ICLONER OVERVIEW

In this section, we give a brief overview on the architecture of iCloner.

2.1 Architecture of the system

Our prototype attack system consists of four main components: The *crawler* component is responsible for crawling the target social networking site and collect information on users that have chosen to make their profiles public. The personal information of these users, therefore, can be accessed by all members of the social network. In some networks, such as Facebook, the default setting does not allow anyone to see any other person's personal information unless they are friends. However, by default, friend lists are public information on Facebook. In contrast, one needs to be friends (i.e., business colleagues) with a person on LinkedIn to be able to see their contacts. Our crawler component is able to crawl through StudiVZ, MeinVZ, Facebook and XING and collect information on contact lists and profiles if these are accessible to the public. The crawler also keeps track of which user profiles could not be retrieved (because of more restrictive user access settings).

The *identity matcher* analyzes the information in the database and tries to identify profiles in different social networks that correspond to the same person. The *profile creator* component can then use this information to create accounts in a social network where the victims have not registered yet, or to duplicate an existing profile inside the same network.

Finally, the *message sender* component is responsible to login into the created accounts and automatically send friend requests to the people that are known to be friends with the victim. Depending on the social networking site that is being targeted, CAPTCHAs might need to be solved to create accounts, to send friend requests, and sometimes even to access a user's profile (if a user sends many requests, a social networking site might request to verify that the user is a real person and not a script).

The CAPTCHAs are analyzed by the *CAPTCHA analysis* component in our system. In particular, we have analyzed the CAPTCHAs that are displayed by SudiVZ, MeinVZ, and Facebook and have designed techniques to break these CAPTCHAs with a success rate that makes automated attacks feasible in practice. Note that we have not encountered CAPTCHAs on LinkedIn, and did not need to solve CAPTCHAs for our experiments with XING.

Figure 1 gives an architectural overview of iCloner and depicts the dependencies between the various components.

2.2 Breaking CAPTCHAs

A CAPTCHA [16] (Completely Automated Public Turing test to tell Computers and Humans Apart) is a type of challenge-response test that is commonly used to determine whether or not the user of a certain application is a human being. Therefore, the key feature of any CAPTCHA algorithm is the ability to generate tests that are at the same time easily solvable by humans, but very hard to solve for a computer application. For instance, since most of the CAPTCHAs are based on the ability to recognized a text in presence of noise, a good CAPTCHA should be resistant against Optical Character Recognition [24] (OCR) techniques.

Just like many other online web services, in social networks, CAPTCHAs are usually employed to prevent automated programs from accessing and abusing the provided services. For example, without CAPTCHAs, it would be trivial for miscreants to crawl the social network in order to automatically collect personal information, and spam the registered users.

Even though breaking CAPTCHAs is not in the focus of this paper, in order to automate our attacks we had to develop a number of CAPTCHA breaking techniques based on a set of open source tools and custom-developed scripts. We used ImageMagick [19] for image filtering, Tesseract [27]

for the text recognition using OCR, and wrote a number of Python and Perl scripts to partition the CAPTCHAs and to apply manipulations at the pixel level.

While far from being highly evolved, our tools are able to break the CAPTCHAs efficiently enough to make automated attacks against the social networking sites StudiVZ, MeinVZ, and Facebook possible. In the following, we briefly describe the techniques we adopted in our CAPTCHA analyzer component.

2.2.1 MeinVZ and StudiVZ CAPTCHAs

Both MeinVZ and StudiVZ require the user to solve CAPTCHAs in order to create new accounts, and to be able to send friend requests. In addition, when the social network is crawled, CAPTCHAs are displayed at regular intervals.

After manually analyzing the CAPTCHAs, we observed that each of them always contains exactly five letters. Each letter is written in a different font, with differing foreground and background colors, and furthermore, it is often tilted, scaled, or blurred. In addition, a simple grid-based noise is added to the image.

We wrote a Perl script to detect and remove the grid noise and to replace the background with white pixels. A second script, then, attempts to identify the connected areas and partitions the image around them in order to isolate the single letters. In case the number of connected regions is different than five (e.g., because two or more letter are partially overlapping), we discard the CAPTCHA and ask the service for a new one. Note that overlappings are not common in the generated CAPTCHAs and in fact, in our experiments we observed overlapping letters in less then 5% of the cases. After the letters in the CAPTCHA are isolated, they are all scaled to the same size and converted to black and white to simplify further analysis.

In the next step, we attempt to match each letter against a set of known fonts. In particular, we compare each font character (tilted from -10 to +10 degrees) against the extracted CAPTCHA letter and count the number of matching pixels. If this number is larger than a certain threshold (dynamically calculated as a percentage of the total number of pixels in the character), we consider the match to be positive.

If the match is not positive, we generate six variations of the unknown letter by applying a chain of ImageMagick's filters (adaptive-blur, contrast, contrast-stretch and black-threshold in different combinations). We then run the Tesseract engine on each variation, and we consider a letter as being recognized if we obtain at least three equal results.

Finally, if we have a positive match for each of the five letters we have extracted from the CAPTCHA, we concatenate them to compose the answer to submit. In MeinVZ and StudiVZ, it is possible to request a new CAPTCHA an arbitrary number of times. However, only three errors are permitted when submitting the answers. Therefore, we adopt a conservative approach and discard all solutions that contain letters that can be easily confused by the text recognition process (e.g., "I" could be confused with "1", "S" could be confused with "5", "0" can be confused with "O", and "8" can be confused with "B").

In our experiments, our technique was not able to recognize all letters in 71% of the CAPTCHAs that were given to us. However, we were able to simply drop the CAPTCHA and ask for a new one. By restricting the analysis to the cases in which our tool was able to match all the letters, our solution was correct 88.7% of the time. That is, considering the fact that both MeinVZ and StudiVZ ban the user after three mistakes, we can solve the CAPTCHA given to us with 99.8% probability in one of the three consecutive attempts.

2.2.2 Facebook CAPTCHAs

Facebook adopts the reCAPTCHA [28] solution. This is a state-of-the-art approach developed at Carnegie Mellon University. The approach consists of using words that are encountered while digitizing books, but that cannot be correctly recognized by the OCR program. Using these words as CAPTCHAs has two main advantages: first, since a computer has failed to recognize them in the first place, they are inherently more difficult to break by automated programs; second, when a human solves a CAPTCHA by reading the words, she contributes to the effort to increase the accuracy of the text of the digitized book.

In order to verify that the user has properly deciphered the CAPTCHA, two words are displayed at the same time. One of the words is an unknown word which the system was unable to read while digitizing a book. The other word is a known word that a number of other users have already been able to identify. When submitting the answer of the CAPTCHA, if the user correctly recognizes the known word, there are good chances that also the answer of the unknown word is correct. In order to make the CAPTCHAs more difficult to break, the words presented to the user are slightly distorted and covered by a curved line.

Unlike the easier CAPTCHAs used by MeinVZ, StudiVZ, or XING, reCAPTCHAs contain meaningful words composed by a variable number of letters. Since partitioning each character would be complicated, and not very helpful, we decided to perform a word-based analysis.

The words displayed by reCAPTCHA are always distorted, both as a consequence of the scanning process, and of the introduction of intentional noise added to complicate the automated text recognition. Therefore, the first step in our technique consists of attempting to unbend the word back to the original shape. In order to do this, our tool extracts the middle line of each word (i.e., the sequence of pixels that are half way between the top and the bottom of each letter). Since this line is very irregular (e.g., it goes up for letters such as "t" and "l" and down for letters such as "g" and "p"), we smooth it by approximating it with a third degree polynomial curve. After this process, we translate each pixel column up or down so that the approximating curve becomes a straight line.

The second phase of our analysis is similar to the one we use for the MeinVZ and StudiVZ CAPTCHAs. We generate a number of different versions of the images containing the CAPTCHA word by applying different combinations of ImageMagick filters. We then run Tesseract on each one.

The text collected from the Tesseract output is then analyzed by a lexical module. First, each word is compared with the content of the English dictionary. If the word we have extracted does not match any known word, the program attempts again with an edit-distance spell correction algorithm [23] to compensate for small errors in the text extraction routine. Unfortunately, a large fraction of the words used in reCAPTCHA are not present in the English dictionary. This is either because these words corresponds to person or geographical names, or because they belong to a different language. Therefore, when the two previous tests

fail, we submit the word to Google. If the number of the Google results is higher then a configurable threshold, we consider the word to be correct. Otherwise, we attempt to substitute it with the Google suggestion, if Google makes one (i.e., we take the word that Google suggests to us with the phrase "Did you mean?:"). If all our tests fail, we drop the CAPTCHA and ask the reCAPTCHA service to give us an alternative one. Note that just like the other CAPTCHAs services we described, probably because of usability issues, reCAPTCHA allows a user to ask for another CAPTCHA if she has problem solving the suggested one.

reCAPTCHA represents the state of the art in CAPTCHA design and, indeed, it is difficult to break on a large scale. In our experiments, we manually verified the result of our system when submitting 2000 reCAPTCHA words. On average, our tool was able to correctly recognize 14% of them. That is, 26% of the CAPTCHAs we submitted correctly identified at least one of the two words.

Unfortunately, from the point of view of the attacker, after a number of failed attempts, reCAPTCHA seems to become more resilient to our CAPTCHA breaking attempts. For example, in the first 100 submitted CAPTCHAs for a given account, the success rate varies between 4% and 7%. That is, we can solve 4 to 7 CAPTCHAs out of a 100 suggested to us. Note that even though the number of submitted answers containing at least one correct word is still between 20% and 30%, our success rate drops in the second hundred CAPTCHAs given to us for the same account. A possible explanation of this is that after a certain number of errors, the system starts to send CAPTCHAs containing *two* known words, thus verifying that both words are recognized correctly.

However, since the number of CAPTCHAs that our attacks require to break is fairly limited (i.e., we need to solve CAPTCHAs only when creating accounts and sometimes when sending friend requests), our attack is still feasible. In fact, Facebook never banned our accounts even after submitting thousands of wrong answers during our experiments. Also, note that an attacker could use a botnet to have access to thousands of different IPs and distribute the CAPTCHA breaking effort among many hosts. For example, if each bot would be deployed to solve only 7 CAPTCHAs per day (i.e., 100 tries), with a botnet consisting of 10,000 bots, the attacker would still be able to send 70,000 friend request messages every day.

3. CLONING ATTACKS

In this section, we present the two automated social engineering attacks that we used in our experiments in order to be able to access the contacts of a victim.

3.1 Profile cloning

Our premise for the profile cloning attack is that social networking users are generally not cautious when accepting friend requests. Our assumption, as an attacker, is that many users will not get suspicious if a friend request comes from someone they know, even if this person is already on their contact list. In fact, some social networking users may have hundreds of confirmed contacts in their friend lists and they may have varying levels of communication with these people. For example, one might exchange messages with a primary school friend once a year, but have daily contact with a friend who is in the same city. Because of the lower degree of contact, the chance that the primary school friend will get suspicious for the new duplicate contact request is less than someone the victim is in regular contact with.

Typically, whenever a user receives a friend request, she needs to confirm the relationship and accept the new connection. Either a standard friendship message can be sent, or a personal message can be added to it. For example, to make the new friend request more convincing, the attacker may add a social engineering message such as "Dear friends, my computer broke down, I am reconstructing my friend list. Please add me again!". While a real attacker would probably use a personal message to increase her success rate, in our experiments, we opted not to add any message to the friend requests in order to evaluate the worst case scenario.

Of course, it is likely that after a while the victims will notice the abnormality in their friend list and will remove the fake friend. Even though this seems to be undesirable, from the attacker's point of view it is enough for a contact to accept a friend request. Even if the contact decides to remove the friend connection later on, the attacker already had a chance to access and copy the victim's personal information.

The profile cloning attack consists of identifying a victim and creating a new account with his real name and photograph inside the same social network. The profile photographs can be simply copied and used when registering a new, cloned account. Furthermore, note that names are not unique on social networks, and people may exist who have identical names.

Once the cloned account has been created, our system can automatically contact the friends of the victim and send friend requests. Whenever a user receives a friend request, she typically sees the photograph and the name of the person who has sent the request. Our expectation, as the attacker, is that the user will accept this request as it is from someone they recognize and know.

Currently, iCloner supports profile cloning attacks on Facebook.

3.2 Cross-site profile cloning

In the cross-site profile cloning attack, our aim is to identify victims who are registered in one social network, but not in another. Our first aim, from the attacker's point of view, is to steal their identities and create accounts for them in the network where they are not registered. Note that this attack is more difficult to detect by the social network service provider or the legitimate owner of the copied profile. As far as the service provider is concerned, a new user is registering to the network.

When creating an identical account in another social network, we try to retrieve as much information as possible from the victim's original account in the other network. Clearly, the type of the social network is relevant when forging accounts. That is, it is much easier for an attacker to create forged accounts in social networks of the same nature. For example, both XING and LinkedIn are focused towards business connections. Therefore, the type of information that users provide in their profiles are of similar nature. Typically, users provide information on their education, their previous jobs, their current jobs and the city and country they live in. The current iCloner implementation is able to automatically compare and forge accounts from XING to LinkedIn.

Our second aim, after the stolen identity has been created, is to identify the friends of the victim in the original network and check which of them are registered in the target network.

To determine with certainty if a friend of the cloned contact is already registered on a different social network is not as straight-forward as it may seem. A simple search for the name of the user may return multiple accounts. As mentioned previously, people might exist on social networks who have identical names as other people. Real names, after all, are known not to be unique. For example, looking for a person with the name "Hans Bauer", a common German name, on Facebook returns 62 different user accounts.

In order to determine with a high probability if a certain user already exists on a social network, we need to look at more information associated with that specific user. We have created a simple scoring system and use a threshold to decide if two accounts with the same name on different networks correspond to the same person. In our comparison, we assign 2 points if the education fields match. According to our experience, it is not very likely that two different users have the same name and the same educational background. If the companies where the users are employed are identical, we assign 2 more points. Finally we assign 1 point if the city and the country where the users are living are also identical. We sum up all the assigned points and if the total score is at least 3, we conclude that the two profiles belong to the same user.

One question we have not discussed yet is how we determine if the information entered by a user in two social networks is identical. In our experiments, we saw that some users may enter differing names for the same type of information. For example, a user may choose to enter "TU Wien" as the educational institution in social network A, and enter "Vienna University of Technology" in social network B. Both entries identify the same institution, but the comparison would fail if we simply compare the two strings. The solution we use is to start a Google search with both terms. Then, we compare the first top three hits. If both terms appear in the top three of the Google search hit list, we decide that both entries are equivalent. Figure 2 depicts the process we use to identify users on two different social networking sites.

Once the contacts of a victim have been identified, our system can then start sending automated friend requests to these identified users. As far as the contacted users are concerned, a friend request is coming from someone who is not on their friend list yet. As a result, our expectation is that most users will accept this request without becoming suspicious. After all, it is the nature of social networks that people get connected by receiving friend requests from time to time from people that they know.

4. EVALUATION

In order to verify the feasibility of our attacks and tools, we conducted real-world experiments with real users. The best way to demonstrate our attacks would have been to start large scale attacks on a large number of real users. However, attacks of this magnitude would have been ethically questionable, and could have lead to legal consequences.

We first started and tested our crawler on two social networks with the aim of collecting large volumes of contact lists and public user profile data. We then performed profile cloning attacks and contacted more than 700 distinct users.

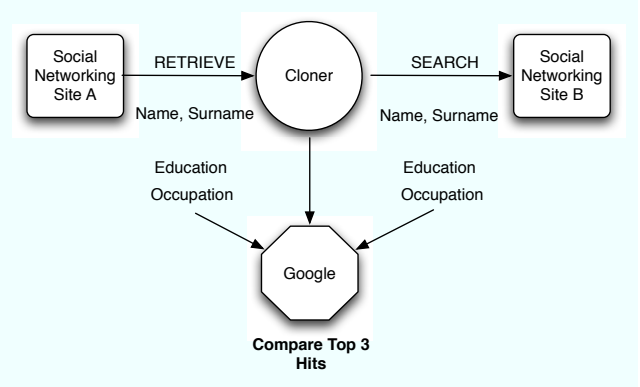

Figure 2: Process used to identify an identical user on two different social networking sites

In a third experiment, we launched cross-site profile cloning attacks and contacted 78 distinct users who were registered on two different social networks.

As we performed our experiments on real user profiles, for each profile we cloned, we requested the permission of the user who was concerned. Furthermore, we committed to making every action we performed transparent to the rightful owners of the profiles. After the experiments, we informed the users who were contacted of our experiment, and also disabled the cloned accounts that we had generated.

4.1 Crawling Experiments

In our crawling experiments, we created a small number of accounts on StudiVZ, MeinVZ, and XING and ran the crawler component of iCloner.

4.1.1 StudiVZ and MeinVZ

First, we created 16 user accounts in these social networks. Then, in order to keep a low profile, we implemented small delays for each page request. For each CAPTCHA request we received, we used our CAPTCHA breaking tools.

Initially, we expected the crawlers to request roughly 100,000 web pages per day, thus retrieving the complete profile information and contact lists of roughly 15.000 accounts (as the contact lists are split and presented in groups of 15 contacts – with an average number of contacts per account being around 100).

Each crawler instance requested and parsed around 6000 web pages per day and encountered on average around 215 CAPTCHAs to break. Because of the little effort needed to break the CAPTCHAs, our crawlers were able to collect information from 40.000 profiles per day, which by far exceeded our expectations.

We stopped the experiment after we had crawled more than 5 million public user profiles with contact information and more than 1.2 million profiles with complete user information.

4.1.2 XING

Our experimental setup for XING was similar to the setup with StudiVZ and MeinVZ.

Interestingly, XING does not contain any CAPTCHA protection in order to prevent automated crawling. However, the service providers were more efficient in disabling accounts that were generating a high number of requests. Nev-

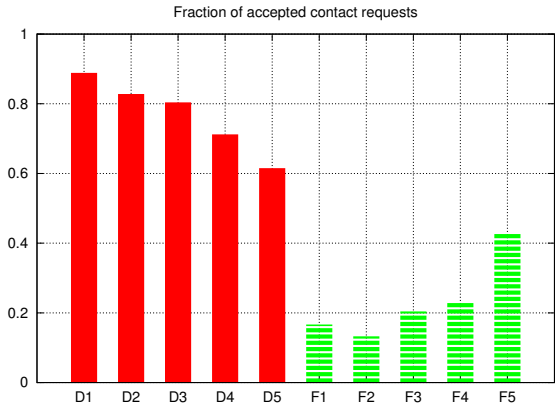

Figure 3: The fraction of accepted contact requests (D1..D5 are the forged profiles, and F1..F5 are the fictitious profiles)

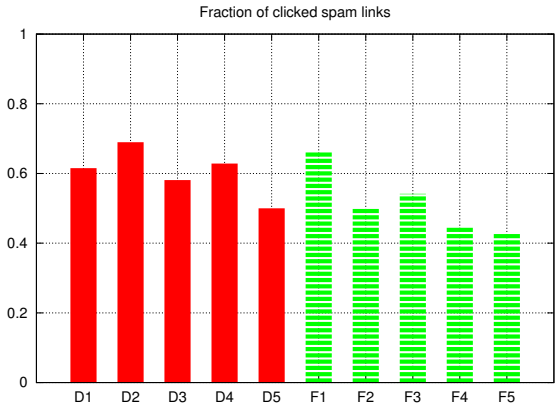

Figure 4: Click through rate for messages sent by forged profiles (D1,..,D5), and fictitious profiles (F1,..,F5)

ertheless, we were able to crawl around 2000 profiles with each account that we generated before our account was disabled. Since the target is to access protected data through cloning and an attacker in this process would constantly create new accounts, this average of accessable profiles per forged account is still more than sufficient.

For XING, we crawled around 118,000 profiles in total, before we stopped the experiment.

4.2 Profile cloning

In the first set of profile cloning experiments we performed, we first wanted to test how willing users would be to accept friendship requests from forged profiles of people who were already on their friendship lists as confirmed contacts. We performed the profile cloning experiments in Facebook.

Using iCloner, we duplicated the profiles of five users (D1,. . . , D5) who had given us their consent for the experiments. For each user, we created a new account with the same name, an arbitrary birth date and the same picture from the original profile. After the profile creation, iCloner sent contact requests to all contacts in the friendship list of each victim. In total, our system contacted 705 distinct users.

In order to measure how effective profile cloning is with respect to requests that the contacted users might receive from people that they do not know, we created a control set of one fictitious profile for each forged profile. These profiles consisted of random names and pictures of arbitrary people. We contacted the same users from these accounts as with the respective forged profiles.

Figure 3 shows the acceptance rate for the forged profiles. Note that the friendship acceptance rate for the forged profiles was over 60% for all the forged accounts (in one case, being as high as 90%). The acceptance rate from unknown users was constantly below 30%, except for one test account that achieved a 40% acceptance rate (this is probably explainable by the fact that this profile belonged to a fictitious, good looking woman). These results confirm that by forging profiles, an attacker can achieve a higher degree of success in establishing contacts with honest users than when using fictitious accounts.

In a second experiment, we wanted to see how much trust users would have in messages that they would receive from their new contacts. To this end, we created a simple non-personal message containing a suspicious link with the text:

```
Hey, I put some more pictures online. Check them
here!:

http://193.55.112.123/userspace/pix?user=<account>
&guest=<contact>&cred=3252kj5kj25kjk325hk}

Ciao, <account first-name>
```

Where `<account>` is the name of the generated account, `<account first-name>` is the first name of the user sending the message, and `<contact>` is the full name of the receiver of the message.

We first sent the link to the users who had accepted contact requests from our fictitious accounts. Then, we sent the request to the contacts of the forged accounts that had not yet received the link, as they had not accepted the contact request from the fictitious profile. In both cases, about 50% of the users who received the link clicked on it. This demonstrates that the attacks we describe can be effectively used for spamming users and directing a large number of users to web sites under the control of the attacker. Interestingly, the difference between the links sent by forged accounts and those sent by the fictitious accounts is not significant as shown in Figure 4.

We also analyzed the delay it took for the receivers to actually click on the links that were sent to them.

Figures 5 and 6 show the Cumulative Density Function (CDF) of the clicks on the links sent by forged accounts and the clicks on the links sent by fictitious accounts. Both graphs show a steep increase during the first ten hours, which subsequently slows down. Hence, about 45% of the users clicked the link in the first 20 hours. Therefore, even though forged accounts would probably be eventually detected and disabled in a large scale attack, the attacker would have enough time to cause damage.

4.3 Cross-site profile cloning

The second profile cloning attack we have performed is a cross-site profile cloning where a profile taken from a so-

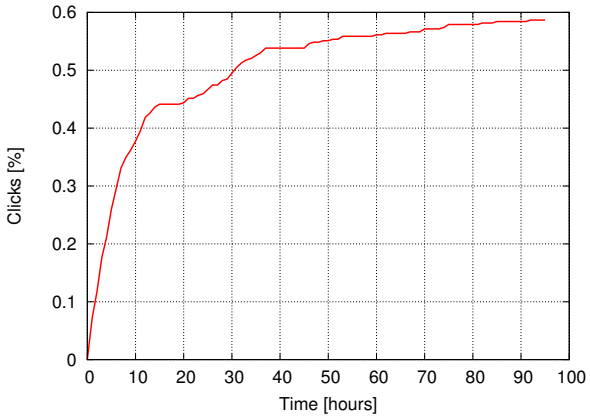

Figure 5: CDF of clicks over time for forged accounts

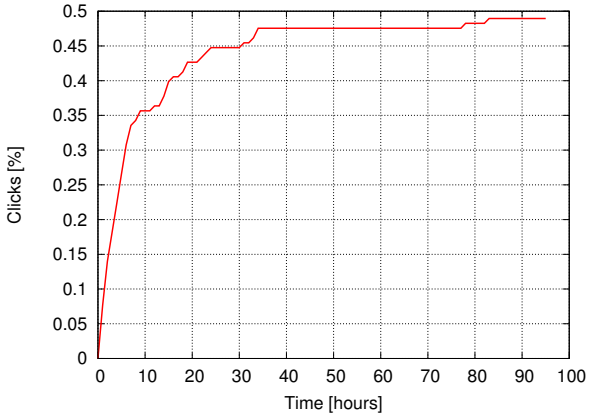

Figure 6: CDF of clicks over time for fictitious accounts

cial network is cloned to another social network. Suppose that the source social netwok is called N_1 and the social network where the profile will be created is called N_2. A profile P in N_1 is chosen to be cloned only if P doesn't exist in N_2 and reasonable amount of P's contacts have profiles in both networks. Therefore, the success of the cross-site profile cloning attack depends on the number of users that have a profile in both the source and the target social network. In our experiment, while the source social network (N_1) is XING, the target social network (N_2) is LinkedIn. In order to have a rough estimate for the success of the attack, we have crawled 30.000 XING profiles and found that 3.700 users (12%) were also registered in LinkedIn. If we take into account that XING has 6.000.000 registered users, the upper bound to number of contacts an attacker can have would be approximately 720.000 which is an attractive number for an attack that aims to perform a massive malicious activity.

Obviously, the user is the most determining factor for the success of the profile cloning attacks. That is to say, the numbers we have estimated above are meaningless if the contacts do not accept the friend requests. To add the user factor to our evaluation, we obtained the consent of 5 XING users to clone their accounts to LinkedIn. iCloner identified that 78 out of their 443 XING (17.6%, cmp. Table 1) friend contacts were also registered on LinkedIn and sent contact

Profiles	LP	SR
X1	18.2%	50.0%
X2	14.5%	66.6%
X3	22.8%	51.6%
X4	14.5%	100.0%
X5	15.6%	46.4%
Total	17.6%	56.4%

Table 1: Percentage of XING profiles found in LinkedIn (LP) and the success rate (SR) of the contact requests

requests to these accounts. Of the 78 contact requests that we sent to the users in LinkedIn, 56%, in total 44, were accepted.

4.4 Discussion

One factor we have not considered in our experiments is what would have happened if the victims whose accounts were being forged would have become suspicious and would have contacted their friends. Clearly, in our experiments, the victims were informed about the experiments we were performing and did not immediately react to warnings by some cautious users that their accounts were probably being cloned. In fact, 4 users sent messages to the original owners of the forged accounts, informing them that something strange was maybe going on. Although such warnings would effect the success of our attacks, we note that most of the users who became suspicious sent warnings *after* they had accepted our contact request. As a result, we were able to access their personal information before they withdrew their authorization.

In most cases, it was interesting for us to see that the users who had added the forged accounts started interacting with the fake accounts as if they were the real ones, and started sending messages and postings.

5. SUGGESTIONS FOR IMPROVEMENTS IN SOCIAL NETWORK SITE SECURITY

Obviously, the user is the weakest link in social networking sites. Many are not security-aware and there is often too much implicit trust. However, even advanced users can be tricked into accepting friend requests as it is not easy to authenticate users. One solution that could improve the security of contact requests would be to provide more information to the receiver on the authenticity of a request and the user who is sending it. For example, the social site could send extra information on where the request was issued (e.g., country information based on the IP) and the profile creation date. Note that sharing this type of information does not pose a privacy threat to users as they are willing to establish a trust relationship, and are therefore willing to share their personal information.

Furthermore, there are simple strategies that can make CAPTCHAs more difficult to break. Our observation is that not all social networking sites put enough effort into making automated crawling and access difficult.

In the CAPTCHAs used by StudiVZ, MeinVZ, and XING, the attacker's main objective is to separate each symbol and subsequently detect each one using OCR. The process of separating could be made more difficult by rendering the image with at least some of the symbols overlapping each

other. In fact, some of the CAPTCHAs that we encountered at these sites are actually rendered with two overlapping symbols, and these are generally not broken. An extension of this strategy is to render additional paths of randomly connected lines spanning over many symbols.

With respect to reCAPTCHA, similar strategies could help strengthen its security. An attacker will generally try to partition the reCaptcha into the two parts and solve each of them separately. Creating an overlap between the two single challenges of the reCaptcha, as discussed before, could again quite easily be done by rendering the paths of random connected lines so as to span over both challenges. reCAPTCHA, without doubt, is a state-of-the-art CAPTCHA solution. Our results show, however, that an attacker who has access to a distributed computing infrastructure (e.g., a botnet consisting of thousands of computers) can distribute the CAPTCHA breaking process over many IPs and machine and succeed in automating access to services that use reCAPTCHAs.

Although straight-forward, rate limiting could make the process of CAPTCHA breaking more difficult. Often, CAPTCHA providers allow the user to request a new CAPTCHA over many attempts and deliver these CAPTCHAs without delay. Hence, it is possible for the attacker to request new CAPTCHAs until the provider delivers a challenge that can be automatically identified and broken. A simple, but effective defense technique would be to rate limit the number of CAPTCHAs that are displayed to a user with a threshold value of a few images per minute.

Finally, social network service providers could adopt (or improve) behavior-based anomaly detection techniques in order to promptly detect and block crawling and other suspicious activities (such as sending hundreds of friend requests in a row). Even though it would still be possible for an attacker to mimic a real user behavior, the attack speed, and therefore its economic viability, would be greatly reduced.

6. RELATED WORK

Social networks comprise of nodes that are connected to each other via strong trusted links. That is, they rely on the assumption that a significant fraction of the users in the system are honest. The most well-known attack to compromise the trust relationship in a social network that employs a reputation system is the *sybil attack* [17]. In this attack, the attacker creates multiple fake identities and pretends to be distinct users in the network, using them to gain a disproportionately large influence on the reputation system.

To date, in order to defend social networks against sybil attacks, two systems were proposed: SybilGuard [29] and SybilLimit [30]. The key insight used in both approaches is that real-world social networks are fast mixing [15, 18] that aids to distinguish the sybil nodes from normal nodes. Fast mixing means that subsets of honest nodes have good connectivity to the rest of the social network.

SybilGuard defines a social network as a graph whose vertices represent users, and whose edges represent the human-established trust relations in the real world. The idea is that if an attacker creates too many sybil nodes and connects them to the network by attack edges, the graph will have a small set of edges whose removal will disconnect a large fraction of the sybil nodes in the network.

Similarly, SybilLimit also assumes and shows that social networks are fast mixing. In comparison to SybilGuard, it ensures more optimal and acceptable limits for the number of sybil nodes in the network.

Both SybilGuard and SybilLimit are good solutions for detecting Sybil nodes. However, in our attacks the established friendship connections are legitimate and the system is establishing contact to a high number of existing "honest" nodes. Therefore, our fake accounts would not be detected by the previous approaches.

A study that is very related to the experiments we present in this paper was conducted by Sophos in 2007 [10]. The authors created a profile on Facebook [4] and manually sent friend requests to 200 random users. The study reports that 41% of the users accepted the request. Furthermore, most of the users did not restrict the access to the personal information in their profile. Note that the results of our experiments are consistent with the study conducted by Sophos and demonstrate that many users are not cautious in social networks. However, one of the main differences between our work and the experiment performed by Sophos is that we are able to automatically identify target users and send friend requests and we show how the attack success rate can be greatly improved by cloning real user accounts.

In [20], the authors present experiments that they have performed on "social phishing". They have crawled a number of social networking sites and have downloaded publicly available information on users. Then, they manually constructed phishing e-mails that contained some personal information on the victims that they were able to retrieve from the social networking sites. The results of the study show that victims are more likely to fall for phishing attempts if some information about their friends or about themselves is included in the phishing mail. Our results, without relying on email messages, confirm that there is a high degree of trust in social networks. However, our focus is different as we aim at accessing the personal information of users that have not necessarily made their profile public.

One of the prerequisites for being able to launch the attacks we present in this paper is the ability to break CAPTCHAs used by a site. Several projects in the area of computer vision exist that provide libraries to break real-world CAPTCHAs (e.g., [22, 26]). Note that our main focus is this paper is not to advance the field of CAPTCHA breaking, but to be able to break the CAPTCHAs efficiently enough to be able to automate the attacks that we describe in the paper. Obviously, some CAPTCHAs are easier to break than others (e.g., StudiVZ and XING are simpler than the reCAPTCHAs employed by Facebook).

To the best of our knowledge, this paper is the first that presents automated cloning attacks against real-world social networking sites and experimentally shows that these attacks are feasible in practice.

7. CONCLUSION

Social networking sites have been increasingly gaining popularity. Many social networking sites have millions of registered users now. Unfortunately, when a new technology starts to attract a large number of Internet users, criminals are attracted as well. Today, it is not uncommon for Internet users to be participants in more than one social networking site (e.g., LinkedIn for business, and Facebook for private networks).

In this paper, we investigate how easy it would be for a potential attacker to launch automated crawling and iden-

tity theft (i.e., cloning) attacks against five popular social networking sites. We present and experimentally evaluate two identity theft attacks. When the attacks succeed, the attacker can establish a friendship connection with the victim's contacts and hence, access their personal information.

The simplest attack we present consists of the cloning of existing user accounts and the automated sending of friend requests to the contacts of the cloned victim. In the second, more advanced attack, we show that it is feasible to launch an automated, cross-site profile cloning attack where the victim's contacts are stolen and reestablished in a social network where she is not registered yet.

We analyzed and experimented with five social networking sites: XING, StudiVZ, MeinVZ, Facebook and LinkedIn. The first three social networking sites are popular in Austria, Germany and Switzerland and have millions of registered users. Facebook and LinkedIn are internationally well-known and also have millions of users world-wide. Our results show that not all social networking sites are well-protected against automated crawling and access. Furthermore, our experimental results demonstrate that most users in social networking sites are not cautious when accepting friend requests or clicking on links that are sent to them.

Although social networking sites are useful, we believe it is important to raise awareness among users about the privacy and security risks that are involved.

8. ACKNOWLEDGEMENTS

This work has been supported in parts by the EU SOCIALNETS project under grant agreement number 217141, the Austrian Science Foundation (FWF) under grant P-18764, the Secure Business Austria (SBA), and the WOMBAT and FORWARD projects funded by the European Commission in the 7th Framework. We thank Michael Roßberg (TU Ilmenau) for fruitful discussions and contributions to the CAPTCHA breaking process and all persons who willing to participate in the experiment.

9. REFERENCES

[1] Modeling and Preventing Phishing Attacks. http://www.informatics.indiana.edu/markus/papers/phishing_jakobsson.pdf, 2005.

[2] Spear phishing: Highly targeted phishing scams. http://www.microsoft.com/protect/yourself/phishing/spear.mspx, 2006.

[3] CERT Advisory CA-2000-04 Love Letter Worm. http://www.cert.org/advisories/CA-2000-04.html, 2008.

[4] Facebook. http://www.facebook.com, 2008.

[5] Facebook by the Numbers. http://www.fastcompany.com/magazine/115/open_features-hacker-dropout-ceo-facebook-numbers.html, 2008.

[6] LinkedIn. http://www.linkedin.com, 2008.

[7] MeinVerzeichnis – MeinVZ. http://www.meinvz.net/, 2008.

[8] MySpace. http://www.myspace.com, 2008.

[9] New MySpace and Facebook Worm Target Social Networks. http://www.darknet.org.uk/2008/08/new-myspace-and-facebook-worm-target-social-networks, 2008.

[10] Sophos Facebook ID Probe. http://www.sophos.com/pressoffice/news/articles/2007/08/facebook.html, 2008.

[11] StudiVerzeichnis – StudVZ. http://www.studivz.net, 2008.

[12] The Spamhaus Project. http://www.spamhaus.org/, 2008.

[13] Xing – Global Networking for Professionals. http://www.xing.com, 2008.

[14] S. D. Berkowitz. *An Introduction to Structural Analysis: The Network Approach to Social Research*. Butterworth, Toronto, ISBN 0409813621, 1982.

[15] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *IEEE INFOCOM*, 2005.

[16] Carnegie Mellon University. The CAPTCHA Project. http://www.captcha.net.

[17] J. R. Douceur. The Sybil Attack. In *Electronic Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, March 2002.

[18] A. D. Flaxman. Expansion and lack thereof in randomly perturbed graphs. Manuscript under submission, 2006.

[19] ImageMagick. Introduction to ImageMagick. http://www.imagemagick.org/script/index.php.

[20] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Commun. ACM*, 50(10):94–100, 2007.

[21] C. Karlberger, G. Bayler, C. Kruegel, and E. Kirda. Exploiting Redundancy in Natural Language to Penetrate Bayesian Spam Filters. In *First USENIX Workshop on Offensive Technologies (WOOT '07), Boston, MA*, August 2007.

[22] kloover.com. Breaking the ASP Security Image Generator. http://www.kloover.com/2008/02/28/breaking-the-asp-security-image-generator/.

[23] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Physics*, 10(8):707–710, 1966.

[24] S. Mori, C. Y. Suen, and K. Yamamoto. Historical review of OCR research and development. *Document image analysis*, pages 244–273, 1995.

[25] S. Moyer and N. Hamiel. Satan is on My Friends List: Attacking Social Networks. http://www.blackhat.com/html/bh-usa-08/bh-usa-08-archive.html, 2008.

[26] PWNtcha. PWNtcha - captcha decoder. http://sam.zoy.org/pwntcha/.

[27] Tesseract. Tesseract OCR. http://sourceforge.net/projects/tesseract-ocr.

[28] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science*, September 2008.

[29] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: Defending Against Sybil Attacks via Social Networks. 2006.

[30] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks. In *IEEE Symposium on Security and Privacy*, 2008.