Enhancing and Identifying Cloning Attacks in Online Social Networks

Zifei Shan, Haowen Cao, Jason Lv, Cong Yan, Annie Liu
Department of Computer Science, Peking University
{shanzifei, haowen.cao, lvx, cong.yan}@pku.edu.cn, fangluliu@gmail.com

ABSTRACT

Recently Online Social Networks (OSNs) are enjoying a continuous boom, while suffering from omnifarious malicious attacks. *Cloning attack* is one of the attack patterns towards online social networks, where typically the attacker disguises fake accounts as real users by thieving and copying their profiles, and sends friend requests to the friends of the cloned victim. It is difficult for ordinary users to detect these fake identities because of the identical names and similar profile information. In this paper, we raise two possible improvements, namely *snowball sampling* and *iteration attack*, to the regular attack pattern upgrading its efficiency and power, so that the attackers can more easily engage into the community. An experiment has been conducted on Renren, the largest OSN in China, to fully compare and substantiate the effectiveness of the enhanced strategy with traditional attacks and different levels of cloning attacks. Then we discuss approaches to detect cloning attacks and put forward a detector named CloneSpotter, which can be deployed into OSN servers. The detector takes advantage of the detailed login IP records and provides solid evidence of locations, in order to judge whether the suspicious accounts are manipulated by real users or attackers. Besides, we discuss a content-based approach to protect users from cloning attacks, which can be easily implemented into distributed clients.

Our contribution lies in two aspects. First, we improve a threatening attack pattern towards OSNs, and test its effectiveness in real systems. Second, we provide an effective defense method to detect cloning attacks, which is real-time and lightweight. By deploying the detectors, OSN systems can assist users to accurately distinguish cloning accounts, and safeguard their privacy.

Keywords

Online social networks, Security, Privacy, Sybil Attack, Identity Theft

1. INTRODUCTION

When you open your Facebook or Twitter, post your new photos and update tweets, your friends can share the photos and make comments. However, this prosperity in OSNs has given rise to the spread of spam and malicious content propagated by many attackers. People want to get rid of these friend requests from spammers and phishers, but sometimes they cannot distinguish them, as each spammers have their ways to disguise themselves.

Traditional attackers merely manipulate multiple random accounts and send quantities of friend requests. These fake identities are easier to be identified by ordinary users since they cannot see any common ground, such as common friends, the same city or the same school, when receiving these friend requests. Another aggressive and oriented attack pattern, known as *cloning attack* or *social phishing*, is to disguise the fake identities as existing people by cloning profiles from real users. This kind of attack is much more dangerous because people get more difficult to distinguish whether the requests are really sent from their acquaintances or from attackers.

In this paper, we enhance the cloning attack pattern with some further strategies, to increase its power and efficiency. An experiment has been launched on a real OSN using the improved attack pattern. In the experiment, we (a) verify the attack's efficiency in a real OSN; (b) analyze the influence of multiple factors in this pattern; (c) compare between the new attack and the traditional pattern. By the experiments we prove that the enhanced cloning pattern is even more dangerous, and can be used in a mass attack.

Then we discuss the categories of methods to detect cloning attacks. We formulate an effective defense method to detect cloning attacks named CloneSpotter. It is a lightweight and high-speed detector deployed in the server side of OSNs. The core idea of CloneSpotter is to judge whether two accounts are controlled by the same person based on comparing a recent list of their login IP addresses. By implementing our detector, the administrators of OSNs can give warnings to normal users and protect them from insecurely accepting friend requests from cloning attackers, and suspend or ban the fake accounts to prevent their further malicious activities. Further discussion is provided about content-based methods in client side to detect cloning attacks, for users to protect themselves.

We have the following contributions to the system: (a) we make improvements on the cloning attack pattern towards

OSNs, and compare the effectiveness between original and improved cloning attacks in real networks. (b) We provide approaches to detect cloning attacks with collaborative detectors, in order to enable OSN systems to assist users distinguishing cloning accounts.

The rest of the paper is structured as the following: Section 2 provides background information on cloning attacks. Section 3 explains our approach to enhance the attack pattern for cloning attacks to make it more powerful towards OSNs. Section 4 presents out the experiment attacking to Renren, the largest OSN in China, to demonstrate the impact of this attack to real systems. Section 5 discusses a fast and efficient real-time detector deployed in the system, along with content-based methods to defend against cloning attacks. Finally we draw a conclusion in Section 6.

2. BACKGROUND

Nowadays, online social networks [5], such as Facebook, Twitter and LinkedIn in US, or Renren and Weibo in China, are facing increasing security and privacy problems. Numerous attackers are propagating spam [8], thieving user data for commercial or malicious usage, and phishing users' password to handle further attacks [9].

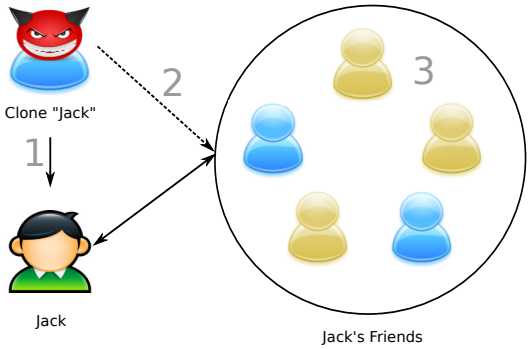
Sybil attacks [4] are one of the most prevalent attack patterns in online social networks. An attacker launching a Sybil attack will manipulate multiple identities to increase its power within a community. [15] Algorithms as SybilGuard [17], Sybil-Limit [16], SybilInfer [3], SybilDefender [13] and SumUp [12] are used to detect Sybil accounts on social networks. These algorithms are mainly based on the assumption that Sybil accounts usually form tight clusters and seldom make friends with normal users. Experiments [14, 6, 10] have been launched on modern OSNs to detect Sybil accounts by analyzing the topology of the network, based on this assumption. However, if the Sybils manage to make many friends with normal users, these algorithms will fail to detect them.

Cloning attack is a new attack pattern, presented by the paper [2]. The attack is the automated identity theft of existing user profiles and sending of friend requests to the contacts of the cloned victim. And the procedure of attack is divided into two main parts.

First, the crawler component will crawl some normal users to get their basic information, including name, city, school, even photos and blogs. In many OSNs, the default settings cannot allow people who are not friends to see each other's personal information, but some basic information and part of friends' list are still available. So the attacker can get this information to establish a *cloning account*, or a *clone* for short, that is a new account registered with some similarity in profile with the original normal user.

Then, the message sender component will use the cloning accounts to automatically send friend requests to the people in the friends' list. Since the cloning accounts' basic information are the same to the people they are familiar with, it is highly possible for these normal users to accept the request.

The procedure of a cloning attack is shown in figure 2.



1. Clone profile
2. Friend requests from attacker
   "I am another ID of Jack!"
3. Some friends (yellow ones) accept

Figure 1: Original cloning attack pattern

The paper [2] has tested the cloning attack on Facebook, which is the largest OSN in the world, and has got the result that this kind of attack has a high percentage of success comparing to the traditional attack models.

In another previous work [11], people have verified that personal profiles can often uniquely identify themselves, and tried to detect cloning attacks by comparing profile similarities.

However, the attack pattern presented by paper [2] is still limited in the following aspects: Firstly, it merely concentrates on the cloned victim's friend list, meaning that this pattern cannot make use of accepted requests to scale the attack to the wider community. Secondly, the attack is scattered and not iterable, therefore it does not give an approach to integrate multiple Sybil users into one specific community, and launch deep and powerful attack towards it.

3. AN ENHANCED ATTACK PATTERN

According to the limitations of the original cloning attack pattern, we make two major improvements to the strategy. *Snowball sampling* technique enables a clone to acquire more friends and be more engaged into a community. *Iteration attack* provides an approach to inject multiple fake identities into a community, thus increases the influence of the attacker among the community.

3.1 Snowball Sampling

Snowball sampling [1] is a general iterative sampling technique. We use its ideas to enhance the original cloning attack pattern. It takes full advantage of the friends who have accepted friend requests sent by cloning accounts. This enhancement is a kind of combination of cloning attack and the traditional attack by typical spammers. After some users agree to be friends to the cloning accounts, the attackers will send requests to people who are these people's friends. Because of the increasing number of existing common friends and the accordent profile data, the clone will soon gain credibility among the community, and acquire a boosting speed on gaining new friends. Therefore, snowball sampling will

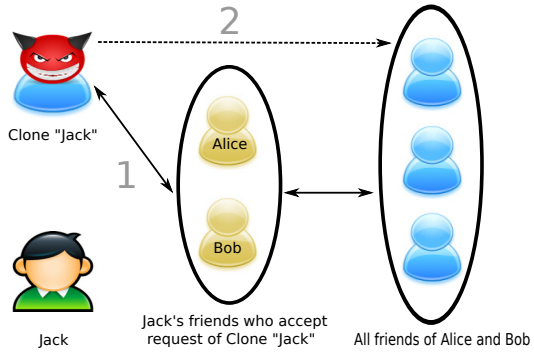


Figure 2: Snowball Sampling

have much better results than simply sending friend request.

For example, the attacker uses Jack's profile to create a "Clone Jack". Then he use Clone Jack to send friend requests to Jack's friends Alice, Bob and Chris. After a while, Alice and Chris accepted the requests. Then Alice's and Chris's friend lists are visible to the attacker, and he can continue to send requests to people in Alice's and Chris's friend lists, say, David. David and Clone Jack have common friends Alice and Chris, so he is prone to accept Clone Jack's friend request. This is an iterative process: after David becomes Clone Jack's friend, Clone Jack can further send requests to David's friends. Figure 2 show this process.

3.2 Iteration Attack

Another kind of enhancement is that once a user accept the request, attackers will have access to more information in detail of this person. Therefore the attacker can register a new cloning account using the detailed profile and even copy photos and blogs to become more "genuine" to others. Consequently, it is difficult for normal people to detect this kind of fake friend requests. The greatest profit for launching an iteration attack is to inject multiple Sybil accounts into a community, so that the attack will gain larger influence in this community, and can use multiple accounts to launch spamming, phishing, advertisements and other activities.

For example, the attacker use Clone Jack to send requests to Jack's friends, and get Alice's acceptance. Then the attacker thieve Alice's detailed profile and created Clone Alice, and use Clone Alice to launch another round of cloning attack to Alice's and Jack's friends. By iteration, the attacker can gradually have multiple fake accounts engaged in the community of Jack and Alice's. Figure 3 show this process.

4. EXPERIMENT: ATTACKING RENREN

To verify the feasibility of our enhanced cloning attack pattern, and compare it with original cloning attacks and traditional Sybil attacks, we conduct real-world experiments with real users on Renren. Renren is the largest online social networks in China, which is known as Facebook's Chinese

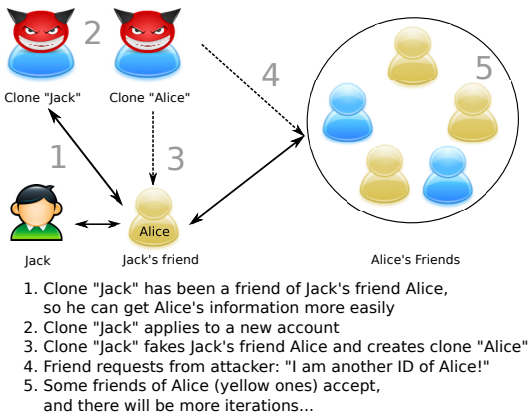


Figure 3: Iteration Attack

tween.

The best way to evaluate the effect of our attack pattern is to launch large scale attacks against many users from different communities. However, due to the complexity and limitation of APIs of automatically creating new accounts, logging in and sending friend requests, we manually create cloning accounts and launch our attack.

4.1 Experiment Methods

In Renren network, there are following contents that can indicate a user's identity: (1) user name; (2) basic profiles, including birthday, school, location, etc; (3) profile photos appearing on the homepage of the user, and (4) blogs, tweets, sharings, etc.

Based on the profile similarity between the *clones*, i.e. cloning accounts, and the *victims*, i.e. original accounts, we classify our clones into three levels: (1) the first level of clones only share the same name with the victims; (2) the second level shares the name, birthday and school; (3) the third level uses more information that can fool others to trust them: we pick a previous profile photo of the original user to be the clones' profile photo. To make comparison of the effectiveness between the cloning attack and traditional attack, we also create accounts with random names.

First we test the original cloning attack pattern. We create three cloning accounts for each level, and all those victims share the same characteristics: they are college students; they are active online; their friend lists each contain about 300 people; and their home pages are visited frequently by their friends. For each victim, we randomly select 24 of their friends and send friend request. To improve authenticity of our cloning accounts, we also send *extra messages* to the requested friends, stating that these accounts has been created by the original person, like: "This is Jack, I have created a new account. Please accept my friend request, thanks!".

Then we test the effectiveness of improved cloning attacks with snowball sampling. In each experiment we first create a cloning account of *level one*, i.e. clones only share the names with the victims. Then we send 12 friend requests to the

victim's randomly selected friends. When user X accepts the request, we continue sending requests to X's friends. The total number of friend requests sending by each clone is still 24.

As control group, we exam the accounts taking the traditional Sybil attack pattern. The accounts with random names send requests to exactly the same targets under the cloning attacks, i.e. the 24 friends for each victim. And no extra messages have been sent under this pattern.

4.2 Results

We count accepted friend requests for each account in twenty-four hours. Table 1 lists experiment results for traditional Sybil attacks, original cloning attacks, and improved cloning attacks, listed in separated columns.

In Table 1, *Incoming requests* are friend requests sent to our clones from friends of the victims whom we didn't send invitation to. An asterisk in *Accepted requests* row means that the clone has been identified by the victims.

4.3 Analysis

From the first experiment, we can see the effectiveness of cloning attack. Compared to traditional attack which only receives 2.7 acceptance out of 24 requests on average, each of the cloning accounts receives at least 5 positive answers for the 24 requests, and even some of them get half of requests accepted, and some other real users take the initiative to send friend requests to the cloning accounts.

According to the results, detailed information of school, city and birthday are important factors that affect the possibility to be accepted as a friend, while the profile photo does not matter much.

The second experiment demonstrates the effectiveness of cloning attack with a snowball sampling, in the level-one profile similarity. In this experiment, 11 requests on average are answered, while 6.3 acceptances have been received by accounts without the snowball sampling optimization. The result proves that snowball sampling obviously increase the chance for cloning accounts to be accepted, as the common friends lead an important role when normal users are dealing with friend requests.

5. DETECTING CLONING ATTACKS

In this section we discuss some methods to detect cloning attacks. Generally, we divide our detecting methods in two major categories. The first is *Content-free Detecting Approach*, which makes judgements according to physical information and actual connections, but neglecting user-generated content. This approach is easy, and it's highly efficient to prevent cloning attacks. The second is *Content-related Detecting Approach*. It checks potential attacker's user content and compares them with the real user's. This approach is less precise, but still it can make valuable reference and can be easily implemented in user's clients.

5.1 CloneSpotter: A Real-time Content-free Detector

In the server side of OSN companies, the detailed data they have can be used to detect cloning accounts. The most simple and effective attribute is the IP address where a user login, and we use this attribute. We name our detector *CloneSpotter*. The detecting algorithm of CloneSpotter is described as Algorithm 1, and explained in the below paragraphs.

ALGORITHM 1. *Detector Architecture*

Input:

U: the set of all users
$Friends_i$: the friend list of user i ($i \in U$)
IPS_i: login IP sequence of user i ($i \in U$)
$Profile_i$: profile data of user i ($i \in U$)
$Similarity_{x,y}$: the similarity of profile x and y ($x, y \in Profile$)

Procedure:

1: ***for all*** *friend requests from A to B ($A, B \in U$)* ***do***
2: $S = \phi$
3: ***for*** $u \in Friends_B$ ***do***
4: ***if*** $u.name = A.name$ ***then***
5: $S = S \cup \{u\}$
6: ***end if***
7: ***end for***
8: ***for*** $u \in S$ ***do***
9: ***if*** $Similarity_{Profile_u, Profile_A} < \lambda$ ***then***
10: $S = S - \{u\}$
11: ***end if***
12: ***if*** $IPS_U \cap IPS_A \neq \phi$ ***then***
13: $S = S - \{u\}$
14: ***end if***
15: ***end for***
16: ***if*** $S = \phi$ ***then***
17: ***return*** *A is not a clone.*
18: ***else***
19: ***return*** *A is a clone of users in S.*
20: ***end if***
21: ***end for***

1. Firstly, for each user, the server records a sequence of distinct prefixes of login IPv4 addresses, say, the most significant 16 bits of IP. Each time a user sign in the website, update the sequence. The sequence records the recently used IPs of users, which indicates their locations when login. The length of the sequence buffer is M (M can be 4). Actually, the sequence of login IP addresses is already stored in Renren server.

2. Secondly, when a friend request is sent by user A to user B, check B's friend list to see whether there is friend of B who has the same name with A. For example, A's name is John Smith. If there is no John smith in B's friend list , then the procedure stops. Otherwise, B has already got at least one "John Smith" in his friend list.

3. Then, for each person u in B's friend list that has the same name with A, compare u's and A's profile and calculate the *similarity* using existing algorithms. Similarity is calculated based on matching genders, schools, living cities and so on. For instance, there is a high possibility that B knows two people whose name are the same, such as John Smith. But actually they are two different people who lives in different cities or

Table 1: Experiment results comparing different attack patterns

Attack Pattern	Traditional			Original Cloning Attack									Snowball Sampling			
				Level 1			Level 2			Level 3			Level 1			
Accepted requests out of 24	3	1	4	6	8	5	8*	12	14	8	18*	7	15	15	8	12
Accepted requests (avg.)	2.7			6.3			11.3			11			12.5			
Incoming requests	0	0	0	0	0	0	0	1	2	0	2	2	2	3	0	2
Incoming requests (avg.)	0			0			1			1.3			1.75			

do different jobs. We should prevent misjudging real users with the same name but is not the same person.

4. If the similarity between A's and u's profiles exceeds a predefined threshold, it means that A's and u's profiles indicate the same person. Still there are two possibilities: A is another account of u, or A is a cloning account created by an attacker who has thieved u's profile. We cannot assert that A is created by the attacker, as perhaps u registers another account to just add good friends and post some private tweets and photos, etc.

5. To judge whether A is a clone, Compare A's and u's login IP sequences. If A and u have joint IP sequences, i.e. there is a same IP prefix in both sequences, then A can be another account of C. Otherwise, A is highly possible to be a cloning account manipulated by an attacker. Actually, there is extremely low possibility for a people having two accounts but the 16-bit IP prefixes have no overlap in the recorded sequence, even if a person have several common places to login the account, such as at home and in the office. Because the server will record the latest M distinct IP addresses, so if it is true that the two accounts are registered by the same person, there will be high probability that at least one same IP address co-occurs in the two sequence. We are also considering the fact that cloning attackers have low possibilities to live in the same place or even city with the victims. So the sequences are likely to be disjoint if A is a clone.

6. Finally, if the account A is detected to be a cloning account created by attackers, the server can give user B a warning to tell him to be cautious, or just suspend or ban account A after it is suspected many times.

The procedure of our algorithm is also demonstrated by a diagram in figure 4.

5.2 System Evaluation of CloneSpotter

In this part we evaluate our CloneSpotter algorithm, and discuss its strengths and drawbacks.

The first strength of our detector is real-time. Real-time detectors are more powerful at preventing the malicious attacks in time, rather than scanning the systems after the Sybils have been widely accepted by users in the community. It can shield the system instantly whenever a cloning attack is launched.

The second benefit is low cost. The detector is featured with a relatively low overhead in both space and time complexity. In terms of storage overhead, the system database shall maintain a distinct login IP prefix sequence for each user. Four slots for the sequence is enough, each slot with 16 bits, thus there will be 64 bits data needed for each user, which is of slight overhead. For systems like Renren, users' login IP addresses is already stored in system, so extra storage is not needed. Considering time complexity, system should add a trigger for each friend request, most of which will halt halfway by the same-name filter, since same-name requests only account for a small portion. The average cost of checking each request is scanning the friend list for a given user name. Suppose the average size of user's friend list is N, the average time complexity will be O(N) for each request, which is tolerable for the system.

However, the detector based on IP address is vulnerable against IP spoofing [7], by which attackers can make clones have same IP with the victims. In this case, we need to dig more into the Content-free information on the server. There are a few possible solutions. (a) Compare the action time pattern of the suspected clone and its target. If they are the same person, they are likely to be active in similar time periods. (b) Compare their click pattern in the users' action log. If they are the same person, they are likely to have similar patterns in visiting the OSNs.

Another potential drawback of the detector is that, we cannot guarantee the detected users to be malicious, in the case that normal users change their IP address and simultaneously change the account, or when normal users change IP suffixes every time. However, with the whitelists in the system, when normal users are misjudged and suspended by the OSN, they can ask the system managers to get their accounts back. Or alternatively, the system can merely use our detector to give warnings to users when they receive friend requests from detected clones, rather than directly suspend them.

5.3 Discussion of Content-related Approach

In most OSN services, one user is not authorized to get other users' physical information — for example the login IP addresses, which means the user can not detect the potential attack according to this sort of Content-free methods, in the case a detector is not deployed in server.

The Content-related Approach makes judgements in a different way. It analyses the potential attacker's content, such as basic profile, status messages, photos, weblogs, and then compare them with real user's.

Basic profile is a user's basic information including birthday, location, schools, etc. Previous work indicates that the basic profile is highly distinctive and the duplication with it can be easily detected. [11] This method is stable, but too objective, so it cannot determine whether the assumed attacker's user

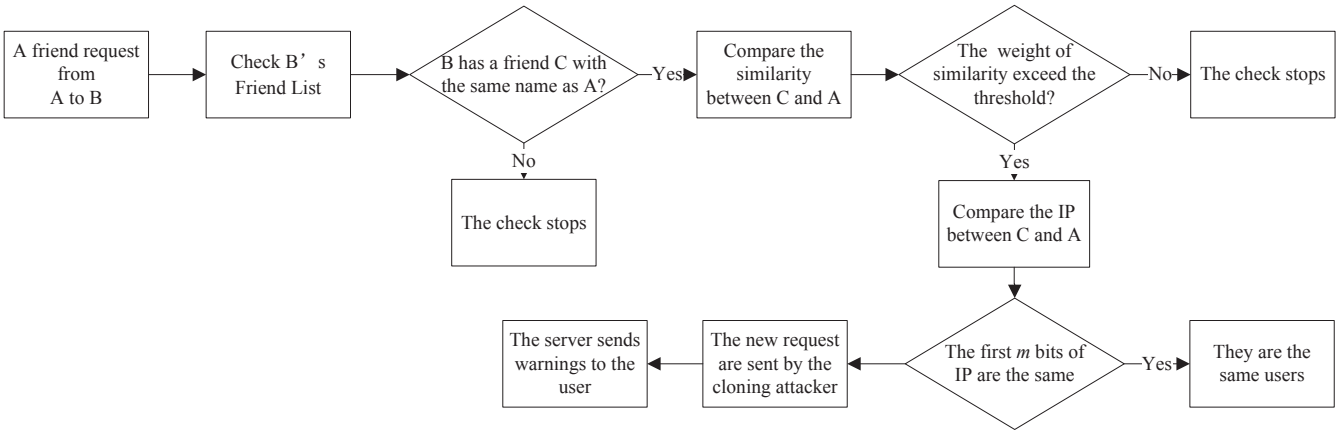


Figure 4: The architecture of system detector

account is a secondary account belonged to the real user himself. Status messages and weblogs comparison is an efficient addition to the method. It checks user's behaviors by semantics, and compare them with real user's. These contents are subjective, so the method is capable to detect whether the account belongs to the real user.

As a matter of fact, the Content-related Approach is highly dependent on user-generated information. So the detection results differ according to various types of attackers. But still, the Content-related Approach has distinctive benefits. The process of the method requires low authority and can be easily experimented. And it can also be implemented as various terminals including web browser extensions and security softwares. Furthermore, the content-based detection leaves more selectivity to users by setting up thresholds in various information detection. This approach can also collaborate with detectors in OSN servers — CloneSpotter for example, to provide the user a highly reliable decision.

6. CONCLUSION

Previous work has been done to define and experiment cloning attack, a kind of Sybil attack towards online social networks. Its overall idea is to thieve the profile data of existing users in the network and create clones disguised as these victims, to easily establish social relationship with the friends of the victims and engage into the community.

We display two effective improvements to the original attack pattern. One is snowball sampling technique that enables a clone identity to acquire more friends and be more engaged into a community; The other is iteration attack, which provides an approach to inject multiple fake identities into a community. Both of these patterns increase the influence of the attacker among the community.

Besides, a real experiment based on Renren has been performed to verify the attack's efficiency and multiple factors which has influence on the experiments has been tested and fully compared. To elaborate the enhanced attack pattern and demonstrate the attacking results in Renren, we hope to raise the vigilance of designers and administrators of OSNs and also normal users. Some effective measures can be taken in both sides to avoid personal information being stolen.

Based on our experimental results, we formulate CloneSpotter, a detector deployed in the server side of OSNs, to give warnings to normal users and assist them in distinguishing these cloning accounts from real people. Important features, including the low-cost of storage in the system database and time-efficiency in the detecting algorithm, has been considered in designing the detector. These two features will enable programmers to easily achieve CloneSpotter to contribute to the robustness of real OSNs' systems. And with the discussions of Content-related Approach, we all believe that it's highly promising to build a distributed and collaborated system in the future work.

7. ACKNOWLEDGMENTS

The authors would like to thank Professor Kaigui Bian, our course advisor, for giving us a lot of encouragement and suggestions on this project, and providing detailed feedback on the composing of paper.

8. REFERENCES

[1] P. Biernacki and D. Waldorf. Snowball sampling: Problems, techniques and chain-referral sampling. *Sociological Methods And Research*, 10(2):141–163, 1981.

[2] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 551–560, New York, NY, USA, 2009. ACM.

[3] G. Danezis and P. Mittal. Sybilinfer: Detecting sybil nodes using social networks.

[4] J. Douceur. The sybil attack. In P. Druschel, F. Kaashoek, and A. Rowstron, editors, *Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer Berlin / Heidelberg, 2002.

[5] L. Garton, C. Haythornthwaite, and B. Wellman. Studying online social networks. *Journal of Computer-Mediated Communication*, 3(1):0–0, 1997.

[6] S. Ghosh, B. Viswanath, F. Kooti, N. K. Sharma,

G. Korlam, F. Benevenuto, N. Ganguly, and K. P. Gummadi. Understanding and combating link farming in the twitter social network. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 61–70, New York, NY, USA, 2012. ACM.

[7] N. Hastings and P. McLean. Tcp/ip spoofing fundamentals. In *Computers and Communications, 1996., Conference Proceedings of the 1996 IEEE Fifteenth Annual International Phoenix Conference on*, pages 218 –224, mar 1996.

[8] P. Heymann, G. Koutrika, and H. Garcia-Molina. Fighting spam on social web sites: A survey of approaches and future challenges. *Internet Computing, IEEE*, 11(6):36 –45, nov.-dec. 2007.

[9] M. Huber, M. Mulazzani, E. Weippl, G. Kitzler, and S. Goluch. Friend-in-the-middle attacks: Exploiting social networking sites for spam. *Internet Computing, IEEE*, 15(3):28 –34, may-june 2011.

[10] J. Jiang, Z. Shan, W. Sha, X. Wang, and Y. Dai. Detecting and validating sybil groups in the wild. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pages 127 –132, june 2012.

[11] G. Kontaxis, I. Polakis, S. Ioannidis, and E. Markatos. Detecting social network profile cloning. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 295 –300, march 2011.

[12] N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, NSDI'09, pages 15–28, Berkeley, CA, USA, 2009. USENIX Association.

[13] W. Wei, F. Xu, C. Tan, and Q. Li. Sybildefender: Defend against sybil attacks in large social networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 1951 –1959, march 2012.

[14] C. Yang, R. Harkreader, J. Zhang, S. Shin, and G. Gu. Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 71–80, New York, NY, USA, 2012. ACM.

[15] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai. Uncovering social network sybils in the wild. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, IMC '11, pages 259–268, New York, NY, USA, 2011. ACM.

[16] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 3 –17, may 2008.

[17] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. *SIGCOMM Comput. Commun. Rev.*, 36(4):267–278, Aug. 2006.