2013

A NOVEL SPAM CAMPAIGN IN ONLINE SOCIAL NETWORKS

Yufeng Zhen
Virginia Commonwealth University

A NOVEL SPAM CAMPAIGN IN ONLINE SOCIAL NETWORKS

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

by

Yufeng Zhen
Master of Science in Computer Application Technology
Jiangsu University, June 2010

Director: Meng Yu, Ph.D.
Associate Professor, Department of Computer Science

Virginia Commonwealth University
Richmond, Virginia
November 2013

ACKNOWLEDGEMENT

Foremost, I would like to express my sincere gratitude to my advisor Dr. Meng Yu for the continuous support of my study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my study. I also want to thank Dr. Wanyu Zang for her help and encouragement. Furthermore, I want to thank all my committee members, including Dr. Hongsheng Zhou, Dr. Wei Zhang and Dr. Meng Yu.

Last but not the least, I would like to thank my parents for giving birth to me at the first place and supporting me spiritually throughout my life.

Contents

List of Figures

Nomenclature

$A_(t)$	The set of targeted accounts
$b_{ij}(p_k)$	Betweenness centrality of node p_k for node pair (p_i, p_j)
$C_B(p_k)$	Overall betweenness centrality of node p_k
$deg(v)$	Number of edges incident with node v
f_{uv}	The friendship value
G	The social graph of OSNs
H	Subgraph of the social graph
n_r	Number of communications our nodes receive
n_s	Number of communications our nodes start
o	The targeted account
$OSNs$	Online Social Networks
r_{ij}	Rate of the transition from current state i to a different state j
s	The socialbot that executes the operation

V	The nodes set of social graph
V_{tn}	The trust value
X_e	The attributes set of the edges
X_n	The attributes set of the nodes
CTMC	Continuous-Time Markov Chains

ABSTRACT

A NOVEL SPAM CAMPAIGN IN ONLINE SOCIAL NETWORKS

By Yufeng Zhen

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2013.

Major Director: Meng Yu, Ph.D.

Associate Professor, Department of Computer Science

The increasing popularity of the Online Social Networks (OSNs) has made the OSNs major targets of spammers. They aim to illegally gather private information from users and spread spam to them. In this paper, we propose a new spam campaign that includes following key steps: creating fake accounts, picking legitimate accounts, forming friendships, earning trust, and spreading spam. The unique part in our spam campaign is the process of earning trust. By using social bots, we significantly lower the cost of earning trust and make it feasible in the real world. By spreading spam at a relatively low speed, we make the life span of our fake accounts much longer than in traditional spam campaigns. This means the trust we have earned can be used multiple times instead of only one time in traditional spam campaigns. We evaluate our spam campaign by comparing it with the traditional campaign, and the comparison shows that our spam campaign is less detectable and more efficient than the traditional campaign.

Chapter 1

Introduction

This chapter covers the background of my research in the area of online social networks(OSNs) and their security-related problems.

1.1 Online social networks

Online social networks (OSNs) such as Facebook, Twitter and Myspace offer services like communicating with other OSNs users and sharing status with others, and are reflections of people's social relationships in the real world. As OSNs have been growing in immense popularity and garnering more and more attention, it is inevitable that spammers focus on them. OSNs provide a way to keep track of our friends' status and to communicate with them more conveniently. Furthermore, OSNs makes it much easier to make new friends, even when they are one thousand miles away.

Take Facebook as an example. Facebook, as a social network service, was established in 2004 [1]. And in nowadays, it has been the most popular online social network in the world, with more than 800 million users all over the world [2]. As one of the OSNs, Facebook requires people

to register to become a user if they want to use Facebook service. And, Facebook encourage users to fill as much information as they can into their profiles, which users can choose to share with with public, their friends or some of them on Facebook.

Facebook users can send friend request to other Facebook users to make friends with them who can be their friends in the real world or total strangers, and others can either deny or confirm the request. And "make friends" here means to create a link between user and another user, which is described as "friendship" by Facebook.

Facebook users can update information they want to share with other users, such as videos, photos and links. And the place they post everything in is called "wall", where users can make it accessible to public, all their friends, some of their friends, friends of their friends or just themselves. And they can also post things on their friend's wall.For users who have not changed the security settings, the default setting is to share their profiles with friends and to share posts with public, which means only friends can see their profiles and every Facebook user can gain access to their posts.

In Facebook, users do accept friend requests from those who are strangers in real world; some of them even send friend requests to strangers to make themselves seem popular. The big difference makes it much more convenient for spammers to spread spam through Facebook.

1.2 Social Graph

The social graph of OSNs is the mapping of people in OSNs and what their relationships are, which contains a graph $G = (V, E)$. It is generated by users' activities: creating accounts, forming friendships with other users, sending messages to friends, and responding to the messages sent by friends. And for every node in V, there is one attribute set X_n belonging to it. Similarly, there is

also one attribute set X_e for every edge in E.

Take Facebook as an example, the social graph of Facebook is a undirected graph. And every node in V represents a Facebook account, and every edge between two nodes in E represents a link between every two accounts. Plus, the attribute set X_n contains name, email, etc. And the attribute set X_e contains the features of the link: friendship, the value of which is taken in $0, 1$, the number of communications our nodes start, and the number of responses we get. Value 1 means there is a friendship between two accounts, and Value 0 means not.

In social graph, creating a node means creating an account, and creating an edge means building a friendship between different accounts.

1.3 Betweenness centrality

Betweenness centrality, according to [3], is a measure of a node's centrality in a network equal to the number of shortest paths from all vertices to all others that pass through that node. $(b_{ij}(p_k))$ is betweenness centrality of node p_k for node pair (p_i, p_j), and it is defined [4] as below:

$$b_{ij}(p_k) = \frac{g_{ij}(p_k)}{g_{ij}} \tag{1.1}$$

In Equation 1.1, p_k represents node k in the graph, p_i and p_j represent two different nodes, g_{ij} represents all the geodesic paths between p_i and p_j.

And, the overall betweenness centrality of node p_k, $C_B(p_k)$ is defined as the following:

$$C_B(p_k) = \sum_{i}^{n} \sum_{j}^{n} \frac{g_{ij}(p_k)}{g_{ij}} \tag{1.2}$$

In Equation 1.2, $1 \leq i < j \leq n$. And, $C_B(p_k) \in [0,1]$. The ideal situation is when $C_B(p_k) = 1$, then we only need make friends with account p_k and post on his/her wall, and all other accounts in the same community can see it. The worst situation is when $C_B(p_k) = 0$, then we have to make friends with all accounts in the same community and post on everyone's wall.

1.4 Vulnerabilities in OSNs defence mechanism

Spammers are those adversaries that try to send spam to other OSNs users. These spam they spread are messages similar to email spam and usually contain malicious links that might lead to malware, phishing attack or different kinds of advertisement.These spam aim to coax receivers to click those links.

To gain as much profit as possible, spammers need a large number of audience to view their spam. Therefore, the tremendous number in OSNs makes users the perfect targets for spammers. To spread spam in OSNs, a spammer has to gain full access to some users in OSNs so that he can send spam to other users. So, a spammer targeting OSNs either has already had or trying to gain full access to some accounts. To fully access accounts in OSNs, a spammer either create fake accounts or compromise legitimate ones. Comparing with compromising legitimate accounts, the advantage of creating fake ones is that the spammer dose not need to worry about the possibility that the original owners of accounts report him to the operator of OSNs. But with full access to compromised accounts, a spammer also has partial access to those friends accounts in OSNs, which means he can spread spam to those friends accounts and doesn't need to endeavor to create fake accounts, form friendships with other ones and earn their trust.

OSNs are vulnerable to spam. It is hard for OSNs to prevent spam from infiltrating, for currently the methods used by OSNs operators, like CAPCHA, valid email address and personal profiles, are not sufficient enough. Furthermore, there are some accounts that still survive after

spreading spam. Without other users reporting spamming accounts to OSNs operators or the number of spread spam reaching the threshold assigned by them, spammers can spread spam without the concern of being detected. Existing defense strategies are based on features they recognized from spam (like spam content, link ratio, and so on) and spamming accounts (like history, posting patterns, and so on). This makes these strategies inflexible, and they are much less effective if spammers change their features.

1.4.1 CAPTCHA

Although OSNs deploy CAPTCHA [5]as a measurement to prevent automatical account creation, an attacker might be able to conquer or bypass this barrier. In [6], the attacker employ KOOBFACE to send panic-causing message including CAPTCHA test to the targeted victims, so they might solve it for the attacker. Similarly in [7], the attacker plant CAPTCHA test into victims' web browsing session without causing their alert, and they might solve the test with no suspicion.In [8], the proposed attack method is able to anaylze and break CAPTCHA. In [9], several attack methods are mentioned, like text recognition, audio recognition, side-channel attacks,feature-based attack, attacks against database-based CAPT-CHAs, and attacks based on implementation and design flaws. In [10], attackers only need to pay very little money to solve CAPTC-HA test, which is usually 0.75 dollar per 1,000 successful cracks of CAPTCHAs, for some working people only 0.5 dollar per 1,000 ones. To save the cost needed on analyzing and implementing attack methods for solving CAPTCHA test, we choose pay for the CAP-TCHA-solving service.

1.4.2 Personal profiles

During the registration of OSNs, the user is always requested to fill his/her personal information, like name, date of birth and sex, which is also how the operator of OSNs prevent the creation of fake

accounts; And more importantly, he/she needs to provide an valid EMAIL address to OSNs, based on which the operator of OSNs is able to detect automated registration that is usually involved in large scale infiltration.

In OSNs, there are two ways for other users to get familiar with you: One is through the communication; The other is by the personal profiles. To every viewer(another user), the user's personal profile is always the first impression. And, a good impression is always a good start to become popular. Thus, to leave a good impression, we need to fill abundant and attractive personal information into the profiles. We can gather pictures of good looking people from internet [8]. And we can fill real person's name into the name field, so it will look authentic to other users.

1.4.3 Decision-making Process

In OSNs, how do operators make the decision of the removal of a spammer account? There are some elements that have to be taken into consideration by them, such as account history, number of friends, communication frequency and user's influence.

The longer history an account has, the less likely it is a spammer one. And if an account spread spam, then its' friends might "unfriend" it. Thus, if an account has relatively more friends, it would seem more convincing to be a legitimate one. If an account barely communicates with any of its friends, that would make it seem suspicious. Therefore, a spammer account should talk with friends to seem normal. And, if an account has major influence on other accounts, surely that will make it harder for operators to decide to remove this account from the network.

Chapter 2

Related Works

In [11], they propose a cross-domain spam detection framework by using associative classification and cross social-corpora classification. To achieve the goal of associative classification, the authors' classifier is based on both the object and respective associated objects. For example, both the message object and the the associated webpage objects are taken into account during the classification. And, the cross social-corpora classification improves the performance of spam detection for the much larger amount of data from across-domain social networks. In [12], they characterize spam campaigns using automated techniques based on the study of wall messages between Facebook users. Based on the study, the authors find that more than 70 percent of malicious wall messages are about luring people to go to phishing websites, and more than 97 percent of spamming accounts are compromised accounts instead of fake accounts. Besides, they find the pattern of spammers' time stamp, which is the early morning, when most of legitimate users are not active. In [13], they use honey pot similar technique to collect data, analyze the collected data and then identify anomalous behavior of users. They create fake accounts, and then collect data passively, which means they accept all friend requests instead of sending any friend requests. By this way, they can reduce the possibility of being reached by legitimate users. In [14] [15], they present

frameworks of Social Honeypot Project for detecting spammers in online social networks. In [16], based on the study of the dataset of million suspended Twitter accounts, they characterize the behavior pattern and lifetime of spamming accounts, the campaigns they execute, and the abuse of legitimate web services. Plus, they detect an emerging marketplace of illegitimate programs operated by spammers. In [17], based on the analysis of spamming accounts in Twitter, they present how to characterize spam and spamming accounts in Twitter. In [18], a spam detecting framework based on user spam reports is presented. The framework is on the basis of HITS web link analysis framework and is implemented in three models that propagate among messages reported by the same user, messages authored by the same user, and messages with similar content. In [19], spam classifiers are developed to detect spam messages spread by collective attention spammers. These spammers exploit users' attention for they easily focus on a phenomenon. In [20], based on the analysis of inner social relationships between criminal accounts and outer social relationships between criminal accounts and their legitimate social friends, a criminal account inference algorithm is presented on Twitter. In [21], a detailed analysis of links, which are acquired by suspended spammer accounts on Twitter, leads to findings that link farming is widely used and most of the farmed links come from a small fraction of Twitter users. In [22], they introduce a realtime immune system that checks and classifies all read and write actions in real time.

In [23], a tutorial and survey on effective fake accounts defenses in online social networks is given. In [24] and [25], they introduce how to identify fake account activities in online social networks. In [24], the authors propose a classification system based on the observation of the difference among human, bot and cyborg in terms of tweeting behavior, tweet content, and account properties.In [25], the authors, based on the study these Sybil accounts on Renren social network, analyze their link creation behavior.

Additionally, in [26], the author finished the creation of fake accounts by using social bots. And this is of great help on my spam campaign.

Chapter 3

Campaign Model

This chapter discusses our approach of our spam campaign. This includes creating spammer accounts , building friendships, earning trust and spreading spam.

3.1 Overview of our approach

Among all the OSNs, we take Facebook as an example to explain our spam campaign. As is shown in Figure 3.1, there are four sequence steps in our spam campaign: spammer accounts creation, building relationships, earning trust, and spreading spam. And to make it much less possible that the spammer accounts are detected by Facebook operators and spread more spam with more clickthrough, we have to follow these steps to finish our spam campaign.

In spammer accounts creation, with the help of socialbots and botmasters, we make large number of fake accounts automatically, through which we spread spam. As the result, we make a huge amount of fake nodes, which belong to set W.

In building relationships, we first form friendships among our fake nodes, which forms edges

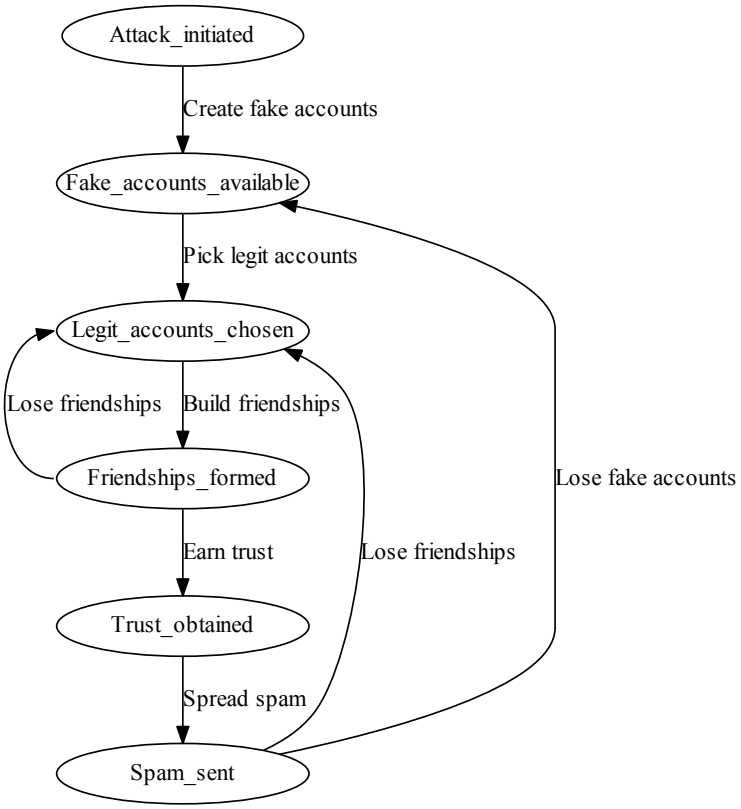

Figure 3.1: State transition graph of the new spam campaign

between our nodes. And the edges are contained in a set F, which, along with set W together, consists a subgraph $H = (W, F)$.

After our subgraph is built, we start building links with legitimate nodes out there in the social graph. And in this way, we add our subgraph into the social graph. And, during this, the links we build between our subgraph and the social graph are the pathes our spam is going to be spread through.

After emerging our subgraph into the social graph, we start the essential step of our campaign, i.e trying to win friends' trust. To achieve this goal, a conversation is initiated with each friend. And a measure of trust, which is the our posts to response ratio, is taken.

Finally, having friends' trust, here comes the final step: spreading spam. The spammer accounts can either directly spread spam(a link for example), or put the spam on the web site and tell friends it is a piece of recently updated deal news, and they would probably go to the web site and click the link. To avoid the operator's attention, we use those large number of spammer accounts we created and make every account spread relatively fewer spams [25].

Furthermore, during the step of building friendships, it is possible that we reach more legitimate accounts for an account can see its friends' friends. Thus, we need redo this step until we have enough friends, or we reach an account that has already been the friend of one of our fake accounts.

Moreover, if we lose friends after spreading spam for some time, we go back to the step of building friendships; and if we lose accounts during spreading spam, we go back to the step of creating spammer accounts.

Continuous-Time Markov Chains(CTMC) [27] is used to describe our spam campaign. CTMC is used to explain a stochastic process with discrete state space. And, in this stochastic process, the present state only depends on the previous state. The matrix in Equation 3.1 is

called infinitesimal generator matrix, which is used to describe CTMC. In Equation 3.1, $r_{ij}(0 \leq i \leq 5, 0 \leq j \leq 5, i \neq j)$ represents the rate of the transition from current state i to a different state j, $r_{ii} = -\sum_{i \neq j} r_{ij}$. If there is a state transition between state i and a different state j, then r_{ij} has a positive value; Otherwise, the value of r_{ij} is zero. r_{ii} is always negative.

$$R = \begin{bmatrix} -r_{0,1} & r_{0,1} & 0 & 0 & 0 & 0 \\ 0 & -r_{1,2} & r_{1,2} & 0 & 0 & 0 \\ 0 & 0 & -r_{2,3} & r_{2,3} & 0 & 0 \\ 0 & 0 & r_{3,2} & -(r_{3,2}+r_{3,4}) & r_{3,4} & 0 \\ 0 & 0 & 0 & 0 & -r_{4,5} & r_{4,5} \\ 0 & r_{5,1} & r_{5,2} & 0 & 0 & -(r_{5,1}+r_{5,2}) \end{bmatrix} \tag{3.1}$$

Chapter 4

Implementation of our spam campaign

In this chapter, I introduce our spam campaign and explain the vulnerability of OSNs. Afterward, I compare the impacts of the RSF intrusion on sensor reputations through various simulations.

4.1 Spammer Accounts Creation

In the whole attack, we create 105 fake accounts, for in [26] about 100 fake accounts have been successfully created. And we form 133 friendships for each of our account, for 130 is the average number of friends in Facebook; and, for the purpose of avoiding that patterns of friends number are recognized by Facebook operators, we will change the number to a random one around 130 for each of our accounts in real attack.

To increase the number of spread spam(N_N), we need as many fake accounts as we can obtain. Thus, we create a large number of fake accounts belonging to the same community which is a real one or at least looks a real one. And every a few of them build friendships between each other, so we can increase every node's degrees. By this way, every node's degree will be greater

than zero, thus, they will seem normal and have higher acceptance rate of friendship requests.

To create an account, we need to provide a valid email address, personal information, besides solving the captcha. And since we need a huge amount of accounts, it would be too much cost if we create them manually. And the social bot would be a excellent choice, for with the help of social bot, we can create accounts automatically.

During the creation, real name should be put in the name field. In [26], the author put photos of good-looking persons collected from internet into fake accounts to increase attractiveness. Besides that, in the profile, we should add abundant information. For example, if we try to spread spam containing football advertisement, then we should fill football related information in the profile, like "I like football.", or "I am a huge fan of NEW YORK GIANTS."

To avoid patterns, we make ten versions of profiles and put there profiles into our accounts. And, we make sure that accounts with same profiles do not make friends with each other or send friend request to the same stranger account. Plus, randomization should be used to make sure that no pattern would exist.

Each one of the accounts we create means adding a new node v into V. With more nodes added into G, we are able to reach more other nodes which represent legitimate users in Facebook. And, before we try to send friendship requests, we form some edges among our nodes, so we build a subgraph $H = (W, F)$.

Socialbot: As mentioned in [26], we need socialbots for accounts we want to create. And as for each socialbot, we need two functions from it. The first one is being capable to communicate with other accounts, which means it is going to be used to read and write, like words and pictures. The second one is that it need be able to form edges in social graph to fit in, i.e. we need it to send friend requests to legitimate accounts and to accept the friend requests from legitimate accounts, if there are any.

Further more, we also define commands, which are to make our account's actions seem like initiated by real a human being instead of a socialbot.

Botmaster: Since we need a large number of accounts, the number of socialbots we need is going to be large,too. Thus, with no doubt, at least one botmaster [26] is needed to control all the socialbots. Plus, for we need different versions of profiles and words(posts) in the conversations with friends accounts, multiple botmasters are needed in our situation, and each of them are in charge of their own accounts.

A botmaster contains three main components: a botworker, a botupdater, and an engine. The botworker creates and maintains socialbots that are under its control. The botupdater keeps socialbots updated with new commands issued by the botmaster. The engine keeps all the commands issued by the botmaster and runs a controller being responsible of sending commands to socialbots when needed.

To reduce the cost of management, some commands are kept in socialbots, while the others are kept in botmaster and are sent to socialbot when needed.

As for the personal profiles needed to fill during the creation, it is going to be better for the safety of our accounts if we make different versions of profiles for every one of them. However, it is not feasible if we consider the cost of creating such large a number of different profiles. Thus, considering the trade-off, we make a compromise which is that we make 10 different versions of profiles for every 10 accounts we create.

4.2 Building Friendships

By forming friendships,the goal is to continue to increase nodes degree, so we can keep those accounts living longer. To achieve this goal and to raise acceptance rate, we try to find those users

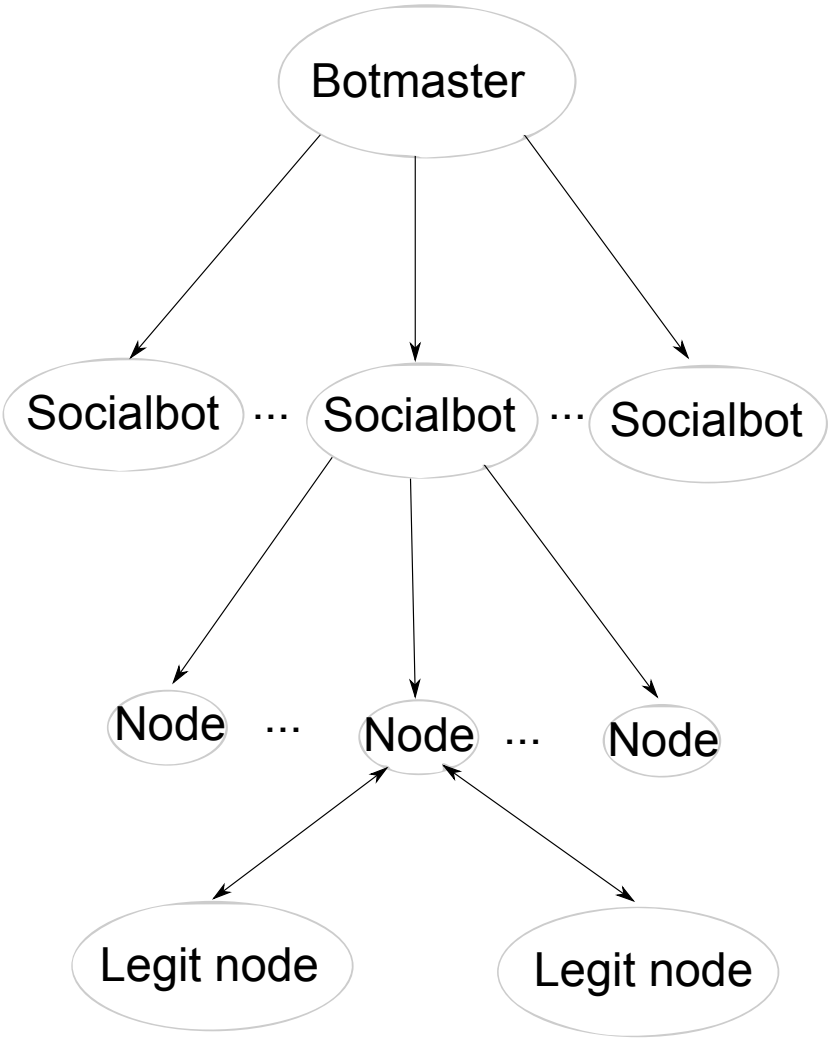

Figure 4.1: Attack Structure of the New Spam Campaign

who want to make friends with any other user no matter if he/she is an acquaintance in real world, or who share the same interest with us. As showed in Figure 4.2, we send friend requests from our fake community to targeted community.

To find who share the same interest with us, we can use method used in [28]. In this paper, using passive de-anonymization method. Authors can breach a large number of users' privacy without the need of being their friends. And, with these private information, we can know what is their profiles, like interest. And this would tell us whom to send friend requests to with the expect of raising acceptance rate.

Besides, we calculate betweenness centrality value of all the nodes in the targeted community.

It would be better if those users have one of these two features: (1) They have a lot of friends, but they only talk to a few of them; (2) They do not have a large number of friends, but they keep sending friend requests. Because these users will more likely accept our friend requests.

Besides that, the spammer accounts also try to make friends with those having the same interest as in written in the profile of spammer accounts.

In Facebook, operators can scan all users and form a social relationship graph, in which those accounts with few and too many friends will stand out. Thus, to make sure our accounts will not stand out in the social relationship graph, we have to try to increase nodes degrees to near the average level of 130. Plus, with more friends, it will be more likely for our friend requests to be accepted.

By the this step, we add the subgraph H into the social graph G. And H is consisted of the accounts that belong to the same community. And adding a subgraph into the social graph has more influence on the social graph than adding individual nodes. That is because with the help a subgraph of large amount of nodes,some of which are connected between each other, we can reach

much more legitimate nodes. Plus, our nodes seem more legitimate within a community, which will help extend every one's life span.

Every friendship request is a try of building a new edge, and every friendship we form represents another edge added into E. And $deg(v)$ is used to denote the number of edges incident with it. Adding $deg(v)$ to the average level will make it much more possible that our spammer accounts nodes do not stand out in the social graph.

We do not intend to let any two of our accounts share a common friend for two reasons. Firstly, it is a waste of increasing cost for two spammer accounts to send spam to the same legitimate account. Secondly, we only have certain number of different versions of profiles, so if a legitimate account is friend of two or more of our accounts, then it is very possible that he/she will find out our accounts are fake.

Plus, as showed in Figure 4.1: We first send an command from the botmaster to the socialbot; And secondly the socialbot initiates the correspondent operation, i.e. it controls its account to carry out the action requested by the operation. And, it always involves our fake account and its friend(s).

Thus, it is absolutely necessary to determine which legitimate accounts we aim to form friendships with before we actually start building friendships. After the decision of targeted legitimate accounts is made , the botmaster sends a command named $friendupdate$ to all of its socialbots, as shown in Figure 4.1. And it is defined as below: After receiving the command $friendupdate$ and the set A_t full of targeted accounts along with it, the socialbot executes the operation $frequest(s,o)$ to all of its targeted accounts, where s is the socialbot that executes the operation, o is the targeted account. After the executtoin of operation $frequest(s,o)$, the socialbot waits for the returned value r_{fr} from every targeted account. And if the targeted this user confirms the friend request, then the returned value r_{fr} should be 1; Otherwise, the returned value r_{fr} should be 0.

And, if $r_{fr} = 1$, push r_{fr} into X_e, which means, in the social graph G, a new edge is formed. At the same time, $deg(v)$, which is initiated as 0, increases by 1.

4.3 Earning Trust

In traditional method, the spammer exploits the trust owned by the legitimate accounts they have hijacked instead of earning it using fake accounts. And there are two reasons for this: Firstly, in traditional method, the spammer needs to log in each of his/her hijacked accounts and send messages to all the friends to earn trust. And the cost of this process is unacceptable. Secondly, traditionally the spammer spreads spam in very short time, so to achieve the goal of spreading as many spam as possible, he/she has to keep the spreading speed high, which will cause the operators' attention and make the life span of their accounts short. And, this means the trust can only be used once during the spam spreading process.

In our method, the way we control the accounts and spread spam causes a much lower cost of earning trust and a much longer life span of our fake accounts, which makes it possible and useful to earn trust before trying to spread spam.

In our method, firstly, the spammer sets up a web site containing some good deal news (link) about what his friends are interested in. Secondly, he keeps updating fresh deal news and tells his friends about it, so that they will go to the web site to look at the updates. Thirdly, our accounts communicate with other legitimate accounts. For example, we share news about recent football game, and we also can make comments on it.

Using this method, spammer accounts can earn other users' trust without the need of posting lots of links, which might get operators' attention.

Let f_{uv} be the friendship value, and n_s be the number of communications our nodes start, and

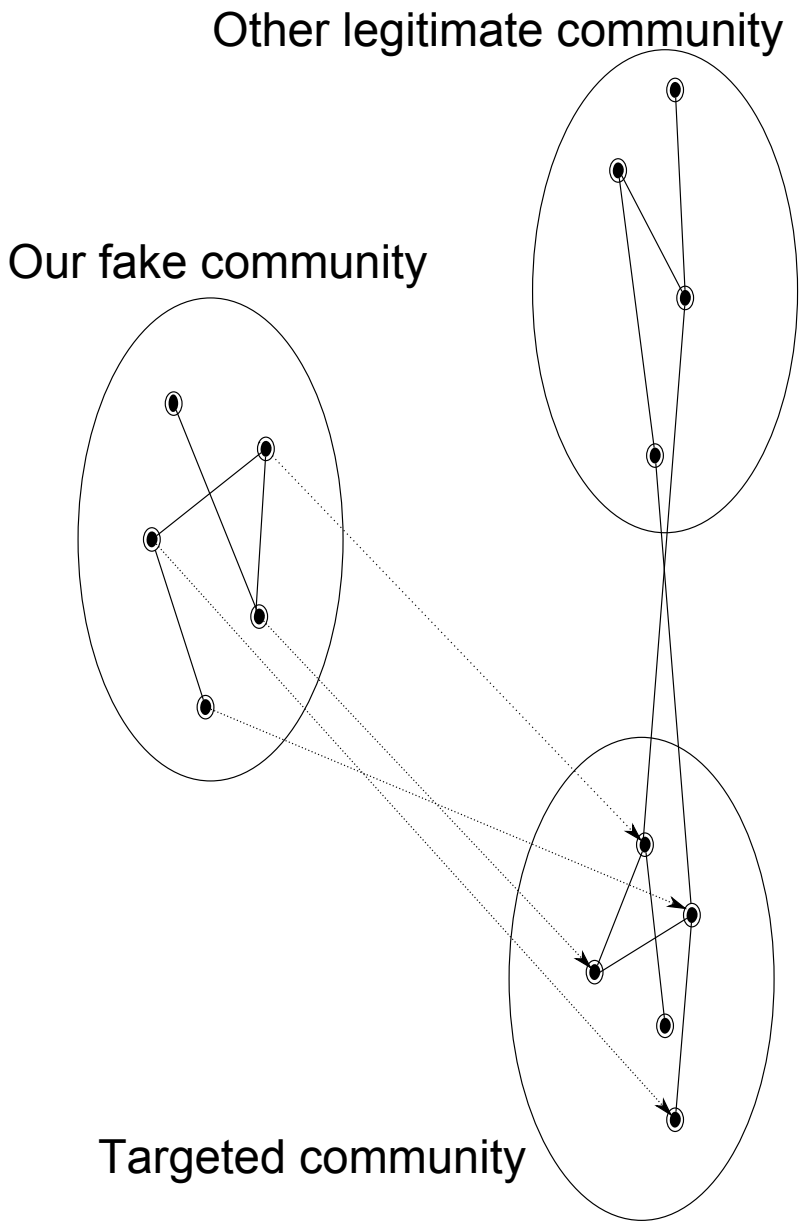

Figure 4.2: Building Relationships

n_r be the number of responses we get. We use V_{tn} which is calculated based on n_s and n_r, which will be mentioned in section 4.6.

When we try to earn friends' trust, that means we try to increase n_s and n_r between nodes u and v, if $f_{uv} = 1$. And, in here, u is one of our nodes who starts the conversation and v is the node that represents the user whose trust we are fighting for.

To communicate with friends accounts, we use a command named $postupdate$ defined as below: when we decide to make a post, each socialbot $b_i \in B$, which received the post update command from its botmater, create a message either crawled from the internet according to our requirements(for example, we require it must be from one specific web, or it must be about a NFL game finished recently, even or it must be about one specific NFL team, etc) or provided directly by us along with that command. And as required by the command, the socialbot will initiate the execution of operation $write(s,o,c)$ to all the friends accounts, where s is the socialbot executes the operation, o is the targeted friend account, and c is the content of message. n_s, which is initiated of the value 0, increases by 1 after every execution of operation $write(s,o,c)$.

After sending a message to a friend account, we will expect the response from him/her. So the botmaster send a command named $detect$ to each of its socialbots. And the command detect is defined as below: every some time, the socialbot executes an operation $check(s,o)$ on its account to see if there has been a response from any of its friends accounts, where s is the friend account from which that response is, o is the socialbot that has been keeping detecting the response. If there is a response, operation $check(s,o)$ will return a signal value 1.

And if a response is detected by the socialbot, it will receive a command named $obtainupdate$ that is automatically sent by itself, which is defined as below: The socialbot b_i execute the operation $read(s,o,c)$, where s is the friend account from which that response is, o is the socialbot that has been keeping detecting the response, and c is the content of message.Furthermore, n_r, which

is initiated of the value 0, increases by 1 after every execution of operation read(s,o,c).

4.4 Spreading Spam

Having friends' trust, the spammer accounts can put the spam(a link for example) on the web site and tell friends it is a piece of recently updated deal news, and they would probably go to the web site and click the link.

It will avoid the operator's attention that we create a large number of spammer accounts and making every account spread relatively fewer spams. It is because the operator always mainly focuses on accounts that spread much more spams.

Let n be a nonnegative integer. A path of length n from u to v in G is a sequence of n edges of G. When one of our accounts spreads spam to a friend, all friends of him/her can see it. So, it means the spam from our nodes can reach any nodes as long as $n \leq 2$. And, it also means when we target a community, instead of spreading spam to all the members, we only need to spread spam to some of them and all of them can see the spam.

After finishing earning trust, we need to do a calculation, and it is defined as following: For each group of friends in one community c_i, the socialbot initiates one null set S_{c_i} and a temporary set filled with all the member nodes of the community, and then calculate the betweenness centrality value $C_B(p_k)$ of all accounts in community c_i, where p_k is each node in the community c_i. After all the calculations are completed, it moves the account with highest betweenness centrality value $C_B(p_k)$ in the same community c_i into the set S_{c_i} and remove all its friends from the temporary set. And, it moves the next available node with highest betweenness centrality value $C_B(p_{k+1})$ into the set S_{c_i} and remove all its friends. It keeps doing this until there are no more nodes in the temporary set. When the execution of command calculate is finished, the socialbot is required to return a

signal of completing the calculation to botmaster.

After receiving the returning signal, the botmaster sends a command named *spam* to the socialbot. The command spam is similar to command post update and is defined as below:

when we decide to spread spam, each socialbot $b_i \in B$, which received the command spam from its botmater, create a message provided directly by us along with that command. And as required by the command, the socialbot will initiate the execution of operation $write(s,o,c)$ to all the friends accounts, where s is the socialbot executes the operation, o is the targeted friend account, and c is the content of the spam message.

As for command post update, each botmaster sends its own version with unique message content to its socialbots. And different from command post update, command spam is exactly the same for all botmasters. That means all socialbots spread the same spam, and this is because the whole spam campaign aims at making as many clickthrough on the spam as possible. And it will lower the clickthrough on each kind of spam if each botmaster sends command spam with different kinds of spam message.

4.5 Our campaign against existing spam detection technologies

In [13], six features are used to detect spammer accounts: FF ratio(R), URL ratio(U), Message similarity(S), Friend choice(F), Message Sent(M) and Friend Number(FN).

FF ratio(R): This feature is the acceptance rate of the friend request. In our method, unlike in the traditional method, we make friends with some of our own accounts first. And, in the first part, the acceptance rate $R_1 = 1$. After this, we continue to build friendships as in traditional method, in this part, the acceptance rate R_2 remains as the same as in traditional method.

Let us assume: In the first part, the number of friend requests is n_f; Similarly, in second part,

the number of friend requests is n_l. Thus, the total acceptance rate in our method R_{new} is defined as below:

$$R_{new} = \frac{n_f R_1 + n_l R_2}{n_f + n_l} \tag{4.1}$$

In Equation 4.1, $R_1 = 1$, thus:

$$R_{new} = \frac{n_f + n_l R_2}{n_f + n_l} \tag{4.2}$$

In traditional method, the acceptance rate R_{old} is defined as below:

$$R_{old} = R_2 \tag{4.3}$$

And, obviously $R_{new} > R_{old}$. Thus, our method does not have this feature.

URL ratio(U): As in [13], it is defined as below:

$$U = \frac{messages_{containingurls}}{total_{messages}} \tag{4.4}$$

In traditional method, most or even all of the messages contain urls, so the URL ratio in traditional method(U_{old}) is very close to 1 or equal to 1. But, in our method, for we use conversations to fight for trust, and we do not put any url in the messages in this period, U_{new} in our method is much lower.

Message similarity(S): Similarly, for we use conversations to fight for trust, and we do not send any similar messages to one friend. Thus, S_{new} in our method is much lower.

Friend choice(F): This feature is to find out a spammer account that uses a name list to pick friend accounts. Since we do not look for friends this way, we do not need to worry about this feature at all.

Message Sent(M): This feature is to find those spammer accounts that only send a small number of messages. And again, since we need send lots of messages in our earn-trust period, the M_{new} in our method is much higher that M_{old} in traditional method.

Friend Number(FN): This feature is that a spammer account has a small number of friends. For we make our friends number FN_{new} to the average level, we do not need to worry about this feature.

In [29], there are 6 six features: Sender social degree, Interaction history, Cluster size, Average time interval, Average URL number per message and Unique URL number.

Sender social degree: The authors in [29] think that the majority of spammer accounts are compromised accounts, so the sender social degree is much higher. For we make our friends numberFN_{new} to the average level, we do not need to worry about this feature.

Interaction history: The authors think a spammer account only communicate with a small part of its friends. Since we send messages to all of our friends accounts while we are trying to earn their trust, we do not need worry about this feature.

Cluster size: This feature is that in every spam campaign the messages sent are similar. But, since we send different messages to our friends while we are trying to earn their trust, we do not need to worry about this feature.

Average time interval: The author thinks that a spammer account spread large number of spam in a very short time. But, since we spread spam after earning trust and we only spread spread a relatively small number of spam every day, we do not need to worry about this feature, either.

Average URL number per message: For the same reason mentioned above, we do not need to worry about this feature, either.

Unique URL number: This feature is based on an assumption that our spam are all blacklissted, but in fact, most of the spam are not in blacklist. Also, since we do not use "embed" to diversify our link into different urls, we do not need to worry about this feature, either.

4.6 Evaluation

In this section, the click rate and number of spread spam are used to compare different spam campaigns.

To measure the click rate, we need to measure how much trust we have earned first. Thus, we use the method based on the frequency friends respond our posts. The more frequently they respond to our posts, the higher the trust value(V_{tn}) is. Similarly, we consider the trust value in traditional method as V_{tt}. And V_{tn} is defined as below:

$$V_{tn} = \frac{n_r}{n_s} \tag{4.5}$$

In Equation (4.5), n_r denotes number of friends' responses, n_s denotes number of our posts, and $V_{tn} \in [0,1]$.

With V_{tn}, we can estimate the click rate. We consider using P_{cn} to denote the click rate for our new method, and P_{ct} for traditional one. The average waiting time(τ) of spam being clicked is defined as below[30]:

$$\tau(V_{tn}) = \frac{1}{(V_{tn})^{\gamma}} \tag{4.6}$$

In Equation (4.6), τ is calculated based on trust(V_{tn}). γ is a parameter [30] that allow us to

interpolate between that random choice limit and the deterministic case. When $\gamma = 0$, V_{tn} does not have any influence on τ, which means the event of spam being clicked happens completely randomly; When $\gamma = 1$, τ is strictly based on V_{tn}, which means the spam sent to an account who has higher trust(V_{tn}) on us is clicked earlier before those with lower V_{tn}.

And, based on Equation (4.6), we can define P_{cn} as below:

$$P_{cn} = \frac{n_(\tau \leq (7 - t_{sent}))}{n_{total}} \tag{4.7}$$

In Equation (4.7), t_{sent} is the day when a message is sent, 7 is that time period of our spreading spam process, $n_(\tau \leq (7 - t_{sent}))$ is the number of messages that received response from friends before our spreading spam process is over, and n_{total} is the total number of spam sent during our spreading spam process. In Equation (4.7), we assume that a recipient account checks our spam the same day or the next day of the day of spam being sent. And, although it is possible that the recipient accounts that did not click the spam will click them after our spreading spam process, the possibility is too low to consider.

For spammers in traditional method do not try to earn friends' trust, naturally, the trust value in traditional method V_{tt} is 0. Therefore, we can not measure V_{tt}. But, P_{ct} is definitely lower than P_{cn}, for spammers in traditional method do not have friends' trust.

For the traditional spam campaign, we consider number of accou-nts(N_{at}), frequency of spreading spam per day(F_t), expected life span(L_t) and number of spam each account spread per time(n_t). Similarly, we consider N_{an} , F_n , L_n and n_n for our new spam campaign.

Thus, the total number of spread spam for traditional spam campaign (N_T) is defined as below:

$$N_T = \sum_{i=1}^{N_{at}} n_t F_t L_i \tag{4.8}$$

In Equation (4.8), $L_i \leq L_t$. And we consider F_t equals 1. This is because that it is mentioned [12] that most spammer accounts spread spam when most legitimate accounts are inactive. And there is only one time interval satisfying this requirement in one day, which is when people are asleep. So I choose to spread spam one time when most legitimate users are asleep and away from their accounts. We also consider L_t equals 4, which is because the fact mentioned in [13], which is that, on Facebook, the average lifespan of a spammer account is four days. And, we need to change L_t to a different value if we use it on another OSN.

And, to calculate the total number of direct spam, we need to consider stage 4 as a fiduciary object, for when we are in stage 4, it means we are going to spread spam until all friends of our every fake account are sent a spam. As about the stage 4 in our spam campaign, there are only two possibilities: we fall into stage 4 or we do not. The possibility of us being in stage 4 is P_4, and the other possibility for not being in stage 4 is $(1-P_4)$. And, when we are in stage 4, the number of directly spread spam is $F_n N_{an} T$; In the other hand, the number is 0. And, according to the definition of expectation in probability, the expected value of total number of spread spam(direct spam) for our new spam campaign (N_N) is calculated by defined as below:

$$N_N = E[X] = (F_n N_{an} T)P_4 + 0(1-P_4) = F_n N_{an} T P_4 \tag{4.9}$$

The number of spread spam is consisted of number of directly spread spam and number of indirectly spread spam. And, the number of directly spread spam is named as *number of direct spam* , which stands for the number of spam that directly sent to the targeted legitimate accounts. Similarly, the number of indirectly spread spam is called *number of indirect spam*. And in this case, when we send a spam to a friend of ours, our friend's friends are also able to see this spam, and the number of clicks caused by this is called *number of indirect spam*.

Figure 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11 and 4.12 are about the change of direct

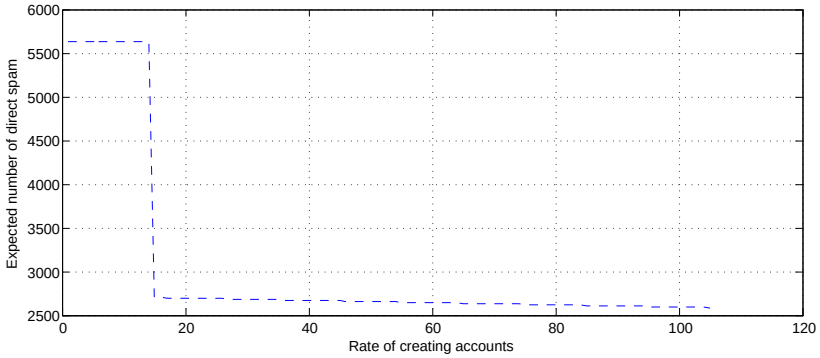

Figure 4.3: Expected number of direct spam with the change of rate of creating accounts

spam number with the change of one or two rates. In these figures, the default values are: $rate\ of\ creating\ accounts = 15$, $rate\ of\ forming\ friendships = 1995$, $rate\ of\ earning\ trust = 1050$, $rate\ of\ spreading\ spam = 805$. And, all these rates change by step of 1. For the convenience of discussion, we assume these values are the threshold values, and any values equal to or above them will draw the operators' attention in a probability. And, the vertical axis *number of direct spam* means the change of it according to the change of one or more rate values.

As shown in Figure 4.3, the changing range of rate *creating accounts*, which is a in Figure 3.1, is from 1 to 105, which is because when a is below 1, it does not make any sense in our attack and we only plan to create 105 fake accounts. There is a threshold about rate a. And, when rate a is lower than or equal to the threshold value, there is no back loop in the attack procedure which means the states go from state 0 to state 5 step by step without going back. We assume the threshold is 15. When rate a is less than 15, the *number of direct spam* stays at 5635, for both rate d and the period of spreading spam do not change. Otherwise, the number changes with rate a, for there is a probability that some of accounts are removed or some of friendships are canceled, and then we need go back to form more friendships or create more accounts. And the probability value goes higher with the increase of rate a. So, when rate a is equal to or larger than 15, the higher it becomes, the smaller *number of direct spam* is.

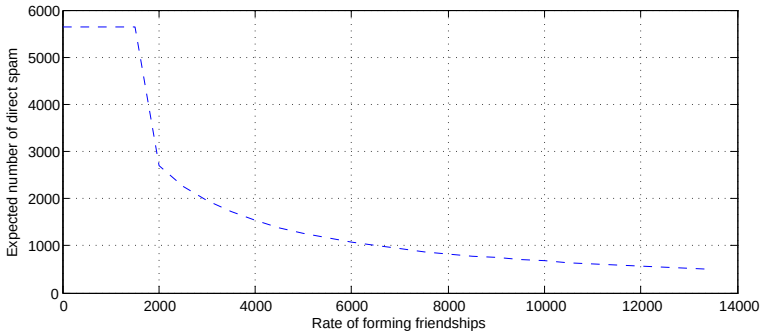

Figure 4.4: Expected number of direct spam with the change of rate of forming friendships

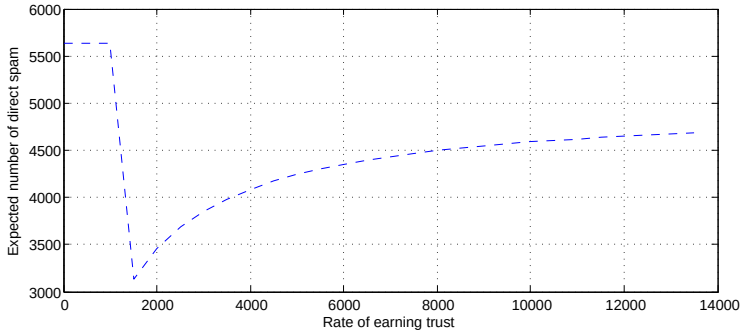

Figure 4.5: Expected number of direct spam with the change of rate earning trust

In Figure 4.4, the changing range of rate $forming\ friend-ships$, which is b in Figure 3.1, b is from 1 to 13965. And, similarly, when rate b is less than 1995, there is no back loop in the attack procedure, and then the $number\ of\ direct\ spam$ stays at 5635. Otherwise, the number decreases with the increase of rate b. Because we need as many as friend accounts to send spam to, and too small or too large number of friends will make our fake accounts stand out in the social graph, we only show the situation that we try to reach around the average number of friends.

In Figure 4.5, the changing range of rate $earning\ trust$, which is c in Figure 3.1, c is from 1 to 13965. And, similarly, when rate c is less than 1050, there is no back loop in the attack procedure, and then the $number\ of\ direct\ spam$ stays at 5635. Otherwise, the number changes with rate c.

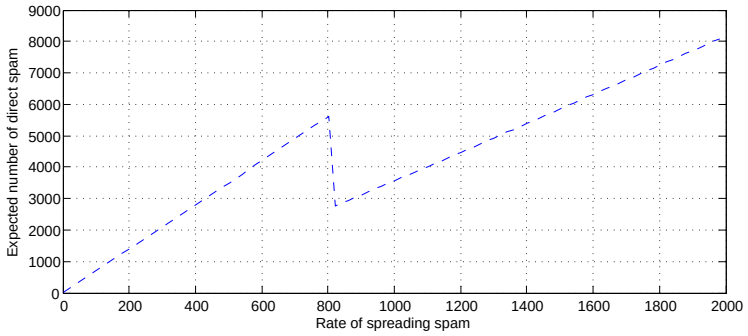

Figure 4.6: Expected number of direct spam with the change of rate spreading spam

And, since c is the rate of earning trust, and in this step, we make normal conversations with friends, even when rate c becomes higher than the threshold, we still can earn trust. And, *spread spam* is the only next step, so the *number of direct spam* goes higher after c passes the threshold. But, when rate c becomes higher than the threshold, which means the frequency of our conversations is higher than normal accounts do, there will be a probability that some of accounts are removed or some of friendships are canceled, and then we need go back to form more friendships or create more accounts. And the probability value goes higher with the increase of rate c. Thus, the *number of direct spam* after c reaches threshold is lower than before.

In Figure 4.6, the changing range of rate *spreading spam*, which is d in Equation 3.1, d is from 1 to 13965. For similar reasons, *number of direct spam* grows faster When rate d is less than threshold 805 than after that. If there is no scan from Facebook operators, the increasing speed of *number of direct spam* stays the same as rate d becomes higher. Unfortunately, for the fact of existence of scans, the higher rate d becomes, the more likely our accounts are detected, which means the probability of being detected goes higher with the increase of rate d. That is why *number of direct spam* grows slower after rate d passes threshold.

In Figure 4.7, when rate a and b are both lower than their own thresholds, the *number of direct spam* stays the same; when a is higher than its own threshold and b is lower than its

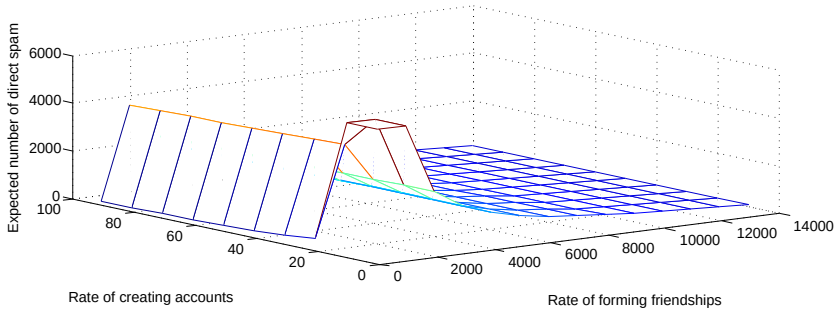

Figure 4.7: Expected number of direct spam with the change of rate of creating accounts(a) and forming friendships(b)

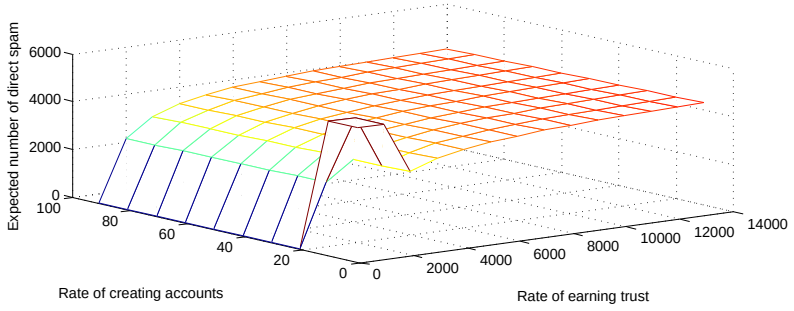

Figure 4.8: Expected number of direct spam with the change of rate of creating accounts(a) and earning trust(c)

own threshold, the *number of direct spam* increase fast; otherwise, the *number of direct spam* decreases.

In Figure 4.8, when rate a and c are both lower than their own thresholds, the *number of direct spam* stays the same; when a is higher than its own threshold and c is lower than its own threshold, the *number of direct spam* increase faster; otherwise, the *number of direct spam* increases lower.

In Figure 4.9, when rate a and d are both lower than their own thresholds, the *number of*

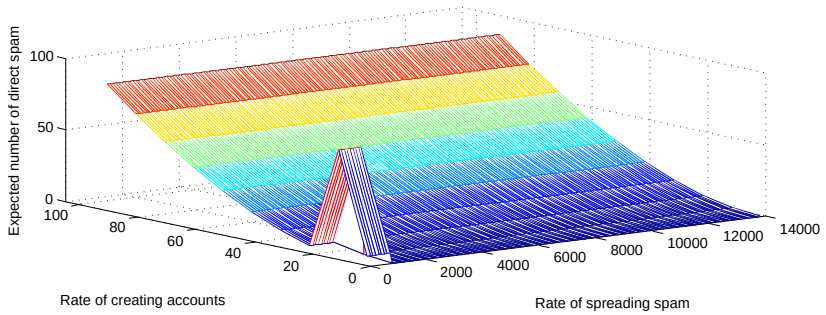

Figure 4.9: Expected number of direct spam with the change of rate of creating accounts(a) and spreading spam(d)

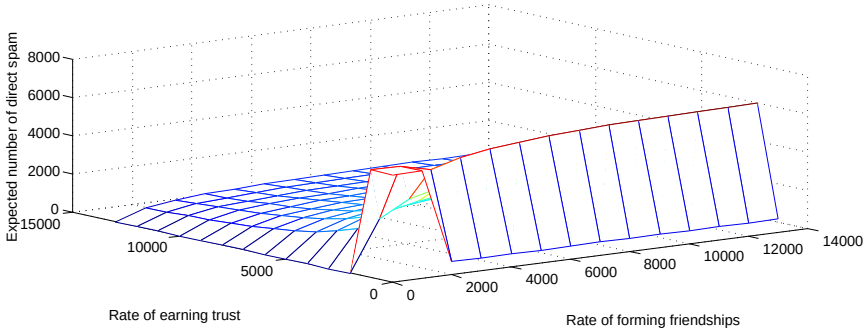

Figure 4.10: Expected number of direct spam with the change of rate forming friendships(b) and earning trust(c)

direct spam increase faster; otherwise, the *number of direct spam* increase relatively lower. But, when d is higher than its own threshold and a is lower than its own threshold, the *number of direct spam* increase lower, and it increases faster with higher a.

In Figure 4.10, when rate b and c are both lower than their own thresholds, the *number of direct spam* stays the same; when c is higher than its own threshold and b is lower than its own threshold, the *number of direct spam* increase faster; otherwise, the *number of direct spam* increases lower.

In Figure 4.11, when rate b and d are both lower than their own thresholds, the *number of*

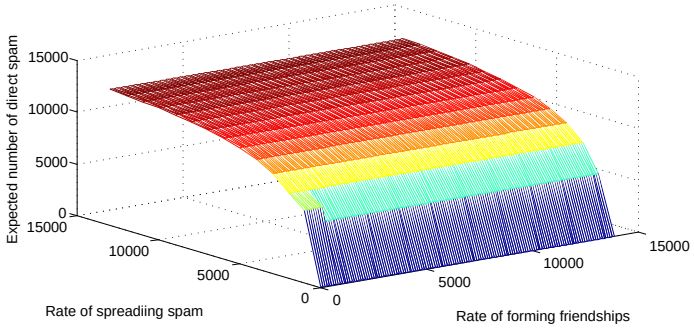

Figure 4.11: Expected number of direct spam with the change of rate forming friendships(b) and spreading spam(d)

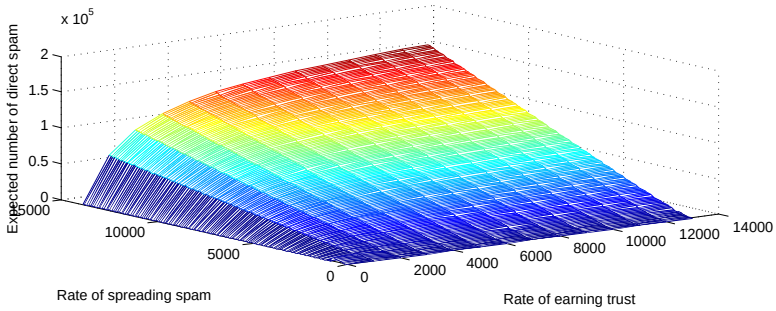

Figure 4.12: Expected number of direct spam with the change of rate earning trust(c) and spreading spam(d)

direct spam increases faster; otherwise, the *number of direct spam* increases lower. But, when b is higher than its own threshold and d is lower than its own threshold, the *number of direct spam* increase relatively faster than they are both higher than their own thresholds.

In Figure 4.12, when rate c and d are both lower than their own thresholds, the *number of direct spam* increases slowest; when only one of c and d is higher than its own threshold, the *number of direct spam* increase relatively faster; otherwise, the *number of direct spam* increases fastest.

In Figure 4.13 and 4.14, the comparison between our new spam campaign and traditional

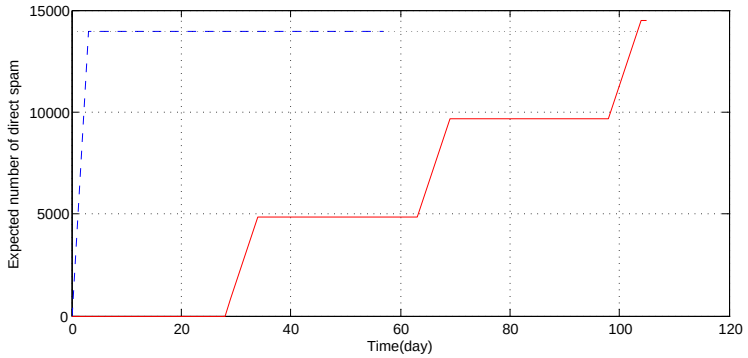

Figure 4.13: Comparison between our new spam campaign and traditional method by repeating whole attack

method is shown. The assumptions of the comparison are in every attack the spam is sent to one account one time, we have the same number of spammer accounts and friends as the traditional method. And the life span of traditional method is three days.

In Figure 4.13, all the attack processes are repeated, and our new campaign exceeds the traditional method during the third repetition. For social network is dynamic instead of keeping static, full repetition of our attack is more consistent with a legitimate user's profile. But, the spreading speed is slower by this full repetition.

But, in Figure 4.14, only last half of the attack processes are repeated, which means we will use the same fake accounts and send spam to the same legitimate accounts. And our new campaign exceeds the traditional method during the second repetition. With higher risk in this repetition, the spreading speed is faster.

And, about the indirect spam, since we choose friends based on the calculation of betweenness centrality value and we choose the ones with higher value, the probability of targeted accounts' friends' clicking on the spam is much higher than that of the traditional method. The action of click is human response and it happens with a time interval, so it is a discretized event. And there must be a distribution that is fit for this, and it will be in the higher part of the distribution

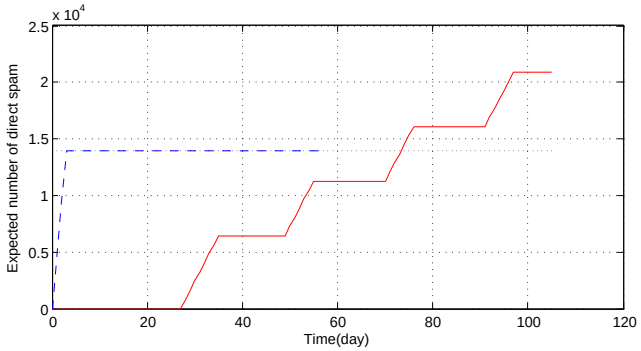

Figure 4.14: Comparison between our new spam campaign and traditional method by repeating last half attack

for our choosing strategy.

Chapter 5

Proposed Defence

Because our spam campaign does not show the same features as traditional one, we need to consider new defence strategy. To simplify the discussion, we assume that there are two reasons for the removal of an spammer account: (1) Facebook users report it to the operators of Facebook; (2) number of spam spread by one account is above the threshold assigned by Facebook immune system.

About the first reason, we think when operators scan their social network, they focus on each account instead of trying to group those accounts that spread the same spam. That is why operators make decisions partially depending on user reports. About the second reason, we think the threshold exists because there are some accounts in Facebook that still survive after spreading spam. In our opinion, their survival is for the number of spam they have spread is below the threshold value and did not stand out during the operators' scan.

With these assumptions, we consider the following defense strategies: (1) group the spammer accounts that are spreading the same spam; (2) group the spammer accounts in the same area.

Chapter 6

Conclusion

In my thesis, a new spam campaign is presented, which is much more ”quite” than the traditional method. That is because the activity pattern of our new spam campaign is more similar to that of the legitimate users. By analyzing the traditional method, we find its shortcomings. And, by changing the activity patterns, we make new spam campaign much less detectable without losing the goal of spreading spam and coax people to click them: By creating fake accounts instead of compromising legitimate accounts, we avoid that other users report about it: By choosing friends before sending friend requests, we raise acceptance rate of the friend requests and increase the number of indirect spam; By earning trust, we raise the click rate of our spam; By spreading spam in a lower speed, we extend the lifespan of our fake accounts, so we do not need to frequently recreate them. Although we do not conduct a real experiment to prove the feasibility and efficiency of our campaign, the result of the simulation still can prove the value of our campaign.

Bibliography

[1] "Bworld robot control software," http://mashable.com/category/facebook/, 2004, [Online; accessed 19-July-2011].

[2] "http://latimesblogs.latimes.com/.../facebook-f8-media-features.html," Los Angeles, CA, USA, 2011.

[3] L. C. Freeman, "A set of measures of centrality based on betweenness," in *Sociometry*, USA, 1977, pp. 35–41.

[4] ——, "Centrality in social networks: Conceptual clarification," in *Social Network*, USA, 1978, pp. 215–239.

[5] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," in *IN PROCEEDINGS OF EUROCRYPT*. Springer-Verlag, 2003, pp. 294–311.

[6] J. C. Jonell Baltazar and R. Flores, "The real face of koobface: The largest web 2.0 botnet explained," in *Trend Micro Research*. Warsaw, Poland: Springer, 2009, pp. 249–311.

[7] E. K. Manuel Egele, Leyla Bilge and C. Kruegel, "Captcha smuggling hijacking web browsing sessions to create captcha farms," in *SAC '10: Proceedings of the 2010 ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2010, pp. 1865–1870.

[8] D. B. Leyla Bilge, Thorsten Strufe and E. Kirda, "All your contacts are belong to us: automated identity theft attacks on social networks," in *WWW '09:Proceedings of the 18th International Conference on World Wide Web*. New York, NY,USA: ACM, 2009, pp. 551–560.

[9] C. J. Hernandez-Castro and A. Ribagorda, "All your contacts are belong to us: automated identity theft attacks on social networks," in *INETSEC 2009 OPEN RESEARCH PROBLEMS IN NETWORK SECURITY*. New York, NY,USA: Springer, 2009, pp. 9–26.

[10] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage, "Re: Captchas: understanding captcha-solving services in an economic context," in *Proceedings of the 19th USENIX conference on Security*, ser. USENIX Security'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 28–28. [Online]. Available: http://dl.acm.org/citation.cfm?id=1929820.1929858

[11] D. Wang, D. Irani, and C. Pu, "A social-spam detection framework," in *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, ser. CEAS '11. New York, NY, USA: ACM, 2011, pp. 46–54. [Online]. Available: http://doi.acm.org/10.1145/2030376.2030382

[12] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao, "Detecting and characterizing social spam campaigns," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 35–47. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879147

[13] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proceedings of the 26th Annual Computer Security Applications Conference*, ser. ACSAC '10. New York, NY, USA: ACM, 2010, pp. 1–9. [Online]. Available: http://doi.acm.org/10.1145/1920261.1920263

[14] K. Lee, J. Caverlee, and S. Webb, "The social honeypot project: protecting online communities from spammers," in *Proceedings of the 19th international conference on World wide web*, ser. WWW '10. New York, NY, USA: ACM, 2010, pp. 1139–1140. [Online]. Available: http://doi.acm.org/10.1145/1772690.1772843

[15] ——, "Uncovering social spammers: social honeypots + machine learning," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '10. New York, NY, USA: ACM, 2010, pp. 435–442. [Online]. Available: http://doi.acm.org/10.1145/1835449.1835522

[16] K. Thomas, C. Grier, D. Song, and V. Paxson, "Suspended accounts in retrospect: an analysis of twitter spam," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '11. New York, NY, USA: ACM, 2011, pp. 243–258. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068840

[17] C. Grier, K. Thomas, V. Paxson, and M. Zhang, "@spam: the underground on 140 characters or less," in *Proceedings of the 17th ACM conference on Computer and communications security*, ser. CCS '10. New York, NY, USA: ACM, 2010, pp. 27–37. [Online]. Available: http://doi.acm.org/10.1145/1866307.1866311

[18] M. Bosma, E. Meij, and W. Weerkamp, "A framework for unsupervised spam detection in social networking sites," in *Proceedings of the 34th European conference on Advances in Information Retrieval*, ser. ECIR'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 364–375. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28997-2_31

[19] K. Lee, J. Caverlee, K. Y. Kamath, and Z. Cheng, "Detecting collective attention spam," in *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality*, ser. WebQuality '12. New York, NY, USA: ACM, 2012, pp. 48–55. [Online]. Available: http://doi.acm.org/10.1145/2184305.2184316

[20] 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*, ser. WWW '12. New York, NY, USA: ACM, 2012, pp. 71–80. [Online]. Available: http://doi.acm.org/10.1145/2187836.2187847

[21] 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*, ser. WWW '12. New York, NY, USA: ACM, 2012, pp. 61–70. [Online]. Available: http://doi.acm.org/10.1145/2187836.2187846

[22] T. Stein, E. Chen, and K. Mangla, "Facebook immune system," in *Proceedings of the 4th Workshop on Social Network Systems*, ser. SNS '11. New York, NY, USA: ACM, 2011, pp. 8:1–8:8. [Online]. Available: http://doi.acm.org/10.1145/1989656.1989664

[23] H. Yu, "Sybil defenses via social networks: a tutorial and survey," *SIGACT News*, vol. 42, no. 3, pp. 80–101, Oct. 2011. [Online]. Available: http://doi.acm.org/10.1145/2034575.2034593

[24] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Who is tweeting on twitter: human, bot, or cyborg?" in *Proceedings of the 26th Annual Computer Security Applications Conference*, ser. ACSAC '10. New York, NY, USA: ACM, 2010, pp. 21–30. [Online]. Available: http://doi.acm.org/10.1145/1920261.1920265

[25] 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*, ser. IMC '11. New York, NY, USA: ACM, 2011, pp. 259–268. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068841

[26] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "The socialbot network: when bots socialize for fame and money," in *Proceedings of the 27th Annual Computer Security Applications Conference*, ser. ACSAC '11. New York, NY, USA: ACM, 2011, pp. 93–102. [Online]. Available: http://doi.acm.org/10.1145/2076732.2076746

[27] B. R. Haverkort, *Markovian Models for Performance and Dependability Evaluation*. Springer Berlin, 2001.

[28] V. S. Arvind Narayanan, "De-anonymizing social networks," in *IEEE Security and Privacy 2009*, USA, 2009.

[29] H. Gao, Y. Chen, K. Lee, D. Palsetia, and A. Choudhary, "Poster: online spam filtering in social networks," in *Proceedings of the 18th ACM conference on Computer and communications security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 769–772. [Online]. Available: http://doi.acm.org/10.1145/2093476.2093489

[30] A.-L. Barabasi, "The origin of bursts and heavy tails in human dynamics," *Nature*, vol. 435, p. 207, 2005. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0505371

Vita

Yufeng Zhen was born on December 17th 1983, in Baoquanling, Hegang city, Heilongjiang province, China. he graduated from No. 1 High School, Baoquanling, Hegang, heilongjiang, China in 2002. He received both his Bachelor and Master degrees in Jiangsu University, China in 2006 and 2010. In 2011, he joined Dr. Meng Yu's security lab as a graduate student.