On the Impact of Social Botnets for Spam Distribution and Digital-influence Manipulation

Jinxue Zhang*, Rui Zhang†, Yanchao Zhang*, and Guanhua Yan‡
*Arizona State University, Tempe, AZ, USA
†University of Hawaii, Honolulu, HI, USA
‡Los Alamos National Laboratory, Los Alamos, NM, USA
*{jxzhang,yczhang}@asu.edu, †ruizhang@hawaii.edu,‡ghyan@lanl.gov

***Abstract*—Online social networks (OSNs) are increasingly threatened by *social bot*s which are software-controlled OSN accounts that mimic human users with malicious intentions. A *social botnet* refers to a group of social bots under the control of a single *botmaster*, which collaborate to conduct malicious behavior, while at the same time mimicking the interactions among normal OSN users to reduce their individual risk of being detected. We demonstrate the effectiveness and advantages of exploiting a social botnet for spam distribution and digital-influence manipulation through real experiments on Twitter and also trace-driven simulations. Our results can help understand the potentially detrimental effects of social botnets and help OSNs improve their bot(net) detection systems.**

I. Introduction

Online social networks (OSNs) are increasingly threatened by *social bot*s which are software-controlled OSN accounts that mimic human users with malicious intentions. For example, according to a May 2012 article in Bloomberg Businessweek,[1] as many as 40% of the accounts on Facebook, Twitter, and other popular OSNs are spammer accounts (or social bots), and about 8% of the messages sent via social pages are spams, approximately twice the volume of six months ago. There have been reports on various attacks based on social bots, such as befriending victims and then grabbing their personal information [1], [2], conducting the spam campaign which leads to phishing, malware, and scams [3]–[5], and conducting political astroturf [6], [7] which refer to campaigns disguised as spontaneous, popular "grassroots" behavior that are actually carried out by a single person or organization.

A *social botnet* refers to a group of social bots under the control of a single *botmaster*, which collaborate to conduct malicious behavior, while at the same time mimicking the interactions among normal OSN users to reduce their individual risk of being detected. For example, social bots on Twitter can follow others and retweet/answer others' tweets. Since a skewed following/followers (FF) ratio is a typical feature for social bots on Twitter [8], maintaining a balanced FF ratio in the social botnet makes it much easier for individual bots to escape detection. Creating a social botnet is also fairly easy due to the open APIs published by OSN providers. For example, we successfully created a social botnet of 1,000 bots on Twitter with $57 to purchase 1,000 Twitter accounts instead of manually creating them.

Despite various studies [9]–[11] confirming the existence of social botnets, the greater danger from social botnets remain to be unveiled. In this paper, we report two new social botnet attacks on Twitter, one of the most popular OSNs with over 500 million users as of 2012 and over 340 million new tweets daily. Our results here can help understand the potentially detrimental effects of social botnets and hopefully help Twitter and other OSNs improve their bot(net) detection systems. Our major contributions are outlined as follows.

- We demonstrate the effectiveness and advantages of exploiting a social botnet for *spam distribution* on Twitter. This attack is motivated by the fact that Twitter will only suspend the accounts that originate spam tweets while not punishing those retweeting spam tweets [12]. If the social botnet is organized as a retweet tree in which only the root originates spam tweets and all the others merely retweet spams, all the social bots except the root bot can escape suspension. Given a set of social bots, we formulate the formation of the retweeting tree as a multi-objective optimization problem to minimize the time taken for a spam tweet to reach a maximum number of victim Twitter users at the lowest cost of the botmaster. Since the optimization is NP-hard, we also give a heuristic solution and confirm its efficacy with real experiments on Twitter and trace-driven simulations.
- We show that a social botnet can easily manipulate the *digital influence* [13], [14] of Twitter users, which has been increasingly used in ad targeting [15], [16], customer-service improvement [17], recruitment [18], and many other occasions. This attack stems from the fact that almost all digital-influence tools such as Klout, Kred, and Retweet Rank, measure a user's digital influence exclusively based on his interactions with others on Twitter. If social bots collaborate to manipulate the interactions of target Twitter users, they could effectively manipulate their digital influence. The efficacy of our attack is confirmed by real experiments on Twitter.
- We point out possible countermeasures which are likely to inspire new important research.

The rest of the paper is organized as follows. §II introduces

[1] http://www.businessweek.com/articles/2012-05-24/likejacking-spammers-hit-social-media

978-1-4799-0895-0/13/$31.00 ©2013 IEEE

the construction of a social botnet on Twitter. §III and §IV show the efficacy and merits of using the social botnet for spam distribution and digital-influence manipulation, respectively. §V discusses the related work. §V concludes this paper and points out possible countermeasures as our future work.

II. Building a Social Botnet on Twitter

We first outline the Twitter basics to help illustrate our work, and the readers familiar with Twitter can skip this paragraph without any loss of continuity. Unlike Facebook, the social relationships on Twitter are unidirectional by users *following* others. If user A follows user B, A is B's *follower*, and B is A's *friend*. In most cases, a user does not need prior consent from another user whom he or she wants to follow. Twitter also allows users to control who can follow them, but this feature is relatively less used. In addition, users can choose to *unfollow* others and *block* their selected followers. A Twitter user can send text-based posts of up to 140 characters, known as *tweet*s, which can be read by all its followers. Tweets can be public (the default setting) and are visible to anyone with or without a Twitter account, and they can also be protected and are only visible to previously approved Twitter followers. A *retweet* is a re-posting of someone else's tweet. A user can retweet the tweets of anyone he or she follows or does not follow, and his retweets can be seen by all his followers. Moreover, a user can *reply* to a post (tweet or retweet) and ensure that specific users can see his posts by *mentioning* them via inserting "@username" for every specific user into his posts. Finally, each user has a *timeline* which shows all the latest tweets, retweets, and replies of his followers.

We construct a social botnet on Twitter consisting of a botmaster and a number of social bots which are legitimate Twitter accounts. Twitter accounts can be manually created or purchased at affordable prices. For example, we bought 1,000 Twitter accounts as social bots with $57 from some Internet sellers for experimental purposes. The botmaster is in the form of a Java application, which we developed from scratch based on the OAuth protocol [19] and open Twitter APIs. It could perform all the Twitter operations on behalf of all social bots to make the bots look like legitimate users. We will show how the botmaster intelligently control the bots for effective spam distribution and digital-influence manipulation in later sections.

III. Social Botnet for Spam Distribution

In this section, we show the advantages and effectiveness of exploiting the social botnet for spam distribution. We start with a motivating example based on real experiments on Twitter in §III-A. Then we formulate spam distribution via the social botnet as a multi-objective optimization problem and present a heuristic solution whose efficacy is evaluated through trace-driven simulations in §III-C.

A. Why the Social Botnet for Spam Distribution?

As the popularity of Twitter rapidly grows, spammers have started to distribute spam tweets which can be broadly defined as unwanted tweets that contains malicious URLs in most cases or occasionally malicious texts [4], [5], [20]. According to a study in 2010 [4], roughly 8% of the URLs in tweets are malicious ones that direct users to scams/malware/phishing sites, and about 0.13% of the spam URLs will be clicked. Given the massive scale of Twitter (over 500 million active users as of 2012 and over 340 million new tweets daily), understanding how spam tweets are distributed is important for designing effective spam-detection mechanisms.

The simplest method for spam distribution is to let social bots distribute spam tweets independently from each other, which we refer to as the *independent method*. In particular, the botmaster can instruct every bot to directly post spam tweets which can be seen by all its followers. According to the Twitter rules,[2] the accounts considered as spam originators will be permanently suspended. Since there are sophisticated techniques such as [21], [22] detecting malicious URLs, this independent approach may subject almost all social bots to permanent suspension in a short time window.

A more advanced method, which we propose and refer to as the *botnet method*, is to exploit the fact that Twitter currently only suspends the originators of spam tweets without punishing their retweeters. In the simplest case, the botmaster forms a single *retweeting tree* , where every bot is associated with a unique vertex and is followed by its children bots. Then only the root bot originates spam tweets, and all the others simply retweet the spam tweets from their respective parent. Given the same set of social bots, both methods can distribute spam tweets to the same set of non-bot Twitter users, but only the root bot will be suspended under the botnet method. Obviously, the botnet method is economically beneficial for the botmaster because it involves non-trivial human effort or money to create a bot account.

We use one experiment on Twitter to validate our conjecture for the independent method. Our experiment uses three different social botnets with each containing 100 bots. The experiment proceeds in hours. At the beginning of every hour, all the bots in the same botnet almost simultaneously post a spam tweet comprising two parts. The first part is different from every bot and randomly selected from the list of tweets returned after querying "music," while the second part is an identical malicious URL randomly selected from the Shalla's blacklists (http://www.shallalist.de/) and shortened using the bitly service (http://bitly.com) for use on Twitter. We find that all the bots in the three botnets are suspected in two, five, and six hours. Based on this experiment, we can safely conjecture that the independent method will cause most bots in a larger botnet to be suspended in a short period, thus putting the botmaster at serious economic disadvantage.

We use a separate experiment to shed light on the merits of the botnet method. In this experiment, we first use 111 bots to build a full 10-ary tree of depth two. The experiment proceed in hourly rounds. At the beginning of every hour of the first round, the root bot posts a spam tweet, while

[2] http://support.twitter.com/articles/18311\#

all its descendants merely retweet the spam tweet after a small random delay. It takes about six hours for the root bot to be suspended, while all the other bots remain alive. Then we replace the suspended bot by a random bot alive, reorganize the bot tree, and starts the next round. We totally run the experiments for five rounds, in each of which only the root bot is suspended after six hours on average. In addition, we test three other different botnets of 2, 40, and 100 bots, respectively, and only the root bot is suspended every time.

It is worth noting that our experiments above focuses on comparing the two methods with regard to the number of suspended bots. How long it takes for Twitter to suspend a bot is not a concern in our experiments, as it depends on when Twitter detects a particular spam tweet and varies due to different malicious URLs embedded in spam tweets.

B. Optimal Social Botnet for Spam Distribution

§III-A motivates the benefits of using the social botnet for spam distribution on Twitter. Given a set of social bots, what is the optimal way for spam distribution? We give an affirmative answer to this important question in this section.

1) **Problem Setting and Performance Metrics***:* We consider a botnet $\mathcal{V}$ of n bots, where each bot $i \in [1, n]$ can be followed by other bots and also other Twitter users outside the botnet (called non-bot followers hereafter). Let $\mathcal{F}_i$ denote the non-bot followers of bot i. Note that $\mathcal{F}_i \cap \mathcal{F}_j$ may be non-empty ($\forall i \neq j$), meaning that any two bots may have overlapping non-bot followers. We further let $\mathcal{F} = \bigcup_{i=1}^{n} \mathcal{F}_i$. How to attract non-bot followers for the bots is related to social engineering [23] and orthogonal to the focus of this paper. Note that it is very easy in practice for a bot to attract many non-bot followers, as shown in [1], [2], [10], [11].

The botmaster distributes spam tweets along one or multiple retweeting trees, and the vertices of every retweeting tree corresponds to a disjoint subset of the n bots. In addition, every bot in a retweeting tree is followed by its children. As discussed, the root of every retweeting tree will originate spam tweets, which will appear in the Twitter timeline of its children bots and then be retweeted. The distribution of a particular spam tweet finishes until all the bots on all the retweeting trees either tweet or retweet it once and only once.

Given a set $\mathcal{V}$ with n bots and $\mathcal{F}$, we propose three metrics to evaluate the efficacy of botnet-based spam distribution.

- *Coverage*: Let $\mathcal{C}$ denote the non-bot receivers of a given spam tweet and be called the *coverage set*. The coverage of spam distribution is then defined as $\frac{|\mathcal{C}|}{|\mathcal{F}|} \in [0, 1]$.
- *Delay*: We define the delay of spam distribution, denoted by τ, as the average time for each user in $\mathcal{C}$ to see a given spam tweet since it is generated by the root bot. A user may follow multiple bots and thus see the same spam tweet multiple times, in which case only the first time is counted.
- *Cost*: We use $|\mathcal{S}|$ and $|\tilde{\mathcal{S}}|$ to define the cost of spam distribution, where $\mathcal{S}$ denote the indices of suspended bots after distributing a given spam, and $\tilde{\mathcal{S}}$ denotes the set of non-bot followers will be lost due to the suspension of $\mathcal{S}$, i.e., $\tilde{\mathcal{S}} = \mathcal{C} \setminus (\bigcup_{i \in \mathcal{V} \setminus \mathcal{S}} \mathcal{F}_i)$.

The above metrics motivates three design objectives. First, we obviously want to maximize the coverage to be one, which happens when all the n bots participate in spam distribution by belonging to one retweeting tree. Second, many malicious URLs in spam tweets are hosted on compromised servers and will be invalidated once detected, and Twitter will remove spam tweets as soon as they are identified. It is thus also important to minimize the delay. Finally, since it incurs non-trivial human effort or money to create bots and attract followers for them, it is critical to minimize the cost as well.

2) **Design Constraints***:* A major design challenge is how to circumvent Twitter's suspension rules[3] that are evolving in accordance with changing user (mis)behavior. We classify the suspension rules into *strict* and *loose* ones. Violators of strict rules will be immediately suspended. The strict rule most relevant to our work is that the users originate spam tweets containing malicious URLs will be suspended. In contrast, a violator of loose rules will initially become suspicious and later be suspended if his violations of related loose rules exceed some unknown threshold Twitter defines and uses internally. Examples of loose rules include repeatedly posting others' tweets as your own or the same tweet, massively following/unfollowing people in a short time period, etc. In addition, the research community have discovered many useful loose rules for spam-tweet detection such as those in [21], [24]–[29] which are likely to be or have been adopted by Twitter into their evolving suspension-rule list. As discussed, we use the botnet method for spam distribution in order to largely circumvent this strict rule. In the following, we introduce five design constraints related to some loose rules we consider most relevant. By following these constraints, the social bots can cause much less suspicion to Twitter and thus are much less likely to be suspended.

1. The maximum height of a retweeting tree is $K = 10$ according to [24]. Hence we claim that any spam tweet will not be retweeted more than 10 times.

2. A bot only retweets the spam tweets posted by its parent bot on the retweeting tree it follows, as retweeting the tweets from non-followed users is known to be effective in detecting spam tweets [27].

3. Any spam tweet from an arbitrary bot will be retweeted by at most $100r$ percent of its followers. As r approaches one, the bot will become increasingly suspicious according to community-based spam detection algorithms [20], [30]. Recall that the followers of any bot $i \in [1, n]$ comprise other bots and also non-bot users (i.e., $\mathcal{F}_i$). Note that non-bot followers rarely retweet spam tweets in practice, but we require all bot followers to retweet spam tweets. Then bot i can have no more than $\lceil \frac{r|\mathcal{F}_i|}{1-r} \rceil$ bot followers.

4. The retweeting lag at any hop $j \in [1, K]$ is a random variable t_i which follows a hop-wise statistical distribution according to [24], as it is quite abnormal for a user to

[3] http://support.twitter.com/articles/18311#

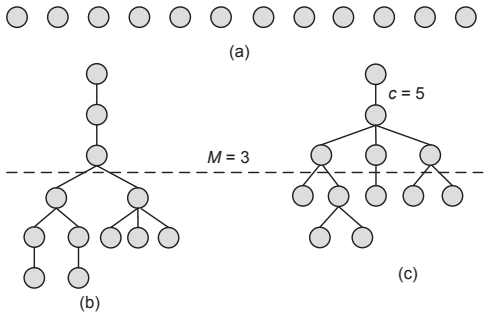


Fig. 1. Exemplary retweeting trees with 12 bots, where $M = 3$ and the botmaster's suspension budget is $c = 5$.

immediately retweet a post once seeing it. Here the retweeting lag is defined as the time elapsed when a bot sees a spam tweet until it retweets it.

5. The social bots within the first M hops will be suspended once Twitter finds that they are involved in (re)tweeting a spam tweet. This constraint is motivated by recent findings [10] that spammer accounts on Twitter tend to be connected and clustered by mutual followings. It is thus reasonable to assume that Twitter either have been utilized or will soon utilize these research findings to suspend the accounts involved in distributing a spam tweet within the first $M > 0$ hops. After introducing this constraint, we relax the third one by allowing arbitrary topology in the first M hops because all of its bots will be suspended.

3) **Problem Formulation***:* Give the above design objectives and constraints, we now attempt to formulate botnet-based spam distribution as an optimization problem. The major challenge lies in the infeasibility of simultaneously achieving the maximum coverage, the minimum delay, and the minimum cost. Fig. 1 shows an example with 12 bots and $M = 3$, and we assume that every bot has the same number of non-bot followers. In one extreme shown in Fig. 1(a), we can minimize the delay τ by letting every bot be a root bot, but the cost is obviously the maximum possible because all the bots will be suspended. In the other extreme shown in Fig. 1(b), we can form a single retweeting tree with exactly three bots within the first three hops, in which case we can achieve the minimum possible cost, but the achievable delay will always be larger than that in the first case no matter how the retweeting tree beyond three hops is formed. In addition, we assume for the second case that the retweeting tree can include all the 12 bots, leading to the same coverage of one as in the first case. If there are too many bots, however, some of them may not be able to be incorporated into the retweeting tree due to the first and third design constraints, and the resulting coverage will be smaller than that of the first case.

To deal with the above challenge, assume that the botmaster has a suspension budget $c \in [M, n]$ bots, referring to the maximum number of suspended bots it can tolerate. Note that the more bots in the first M hops, the more non-bot followers in $\mathcal{F}$ closer to the root bot which can receive a given spam tweet in shorter time, and thus the smaller the delay. Under the budget constraint, the minimum delay can hence be achieved only when there are exactly c bots within the first M hops, as shown in Fig. 1(c) with $c = 5$.

What is the optimal way to form a retweeting tree as in Fig. 1(c) given the cost, coverage, and delay requirements? Recall that the cost is defined by $|\mathcal{S}|$ and $|\tilde{\mathcal{S}}|$. Since $|\mathcal{S}| = c$ under the budget constraint, we just need to minimize $|\tilde{\mathcal{S}}|$. To mathematically express the cost and coverage requirements, we let $\{\mathcal{V}_k\}_{k=1}^{K}$ denote K disjoint subsets of the bot indices $\{1, \ldots, n\}$, where $K = 10$ is the maximum height of the retweeting tree (see Constraint 1), $\mathcal{V}_k$ denote the bot indices at level k of the retweeting tree, and $\bigcup_{k=1}^{K} \mathcal{V}_i \subseteq \{1, \ldots, n\}$. If the optimal retweeting tree eventually found is of depth $K^* < K$, the sets $\{\mathcal{V}_k\}_{k=K^*+1}^{K}$ will all be empty. Recall that $\mathcal{F}_i$ denotes the set of non-bot followers of bot $i \in [1, n]$ and that $\mathcal{F} = \bigcup_{i=1}^{n} \mathcal{F}_i$. Then we have $\tilde{\mathcal{S}} = \mathcal{C} \setminus (\bigcup_{i \in \mathcal{V}_k, k \in [M+1, K^*]} \mathcal{F}_i)$ and the coverage set $\mathcal{C} = \bigcup_{i \in \mathcal{V}_k, k \in [1,K]} \mathcal{F}_i \subseteq \mathcal{F}$ and need to maximize $|\mathcal{C}|$. Since $\tilde{\mathcal{S}} \subseteq \mathcal{C}$, we can combine the cost and coverage requirements into a single metric $\frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|}$ and then attempt to minimize it.

It is a little more complicated to derive the delay. As discussed, a non-bot user may follow multiple bots at different levels, in which case it is considered a follower on the lowest level among those. Let Φ_k denotes the set of non-bot followers at $k \in [1, K]$. It follows that $\Phi_1 = \mathcal{F}_i$ $(i \in \mathcal{V}_1)$ and $\Phi_k = \bigcup_{i \in \mathcal{V}_k} \mathcal{F}_i - \bigcup_{l=1}^{k-1} \Phi_l$ for $k \in [1, K]$. According to Constraint 4, it takes $\sum_{j=1}^{k} t_j$ for a spam tweet to reach level-k non-bot followers, where t_j denotes the retweeting lag of hop $j \in [1, K]$, and $t_1 = 0$. Since there are totally $|\Phi_k|$ non-bot followers at level k and $|\mathcal{C}|$ non-bot followers across all levels, we can compute the delay as

$$\tau = \frac{1}{|\mathcal{C}|} \sum_{k=1}^{K} \sum_{j=1}^{k} t_j |\Phi_k| \ .$$

Finally, we reduce the three-objective optimization problem to the following single-objective minimization problem.

$$\begin{aligned} \min \quad & f(\{\mathcal{V}_k\}_{k=1}^{K}) = \alpha\beta \frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|} + (1-\alpha)\tau \\ \text{s.t.} \quad & \bigcup_{k=1}^{K} \mathcal{V}_i \subseteq \{1, \ldots, n\} \\ & \mathcal{V}_i \cap \mathcal{V}_j = \phi, \forall i \neq j \in [1, K] \\ & |\bigcup_{k=1}^{M} \mathcal{V}_i| \leq c \\ & \sum_{i \in \mathcal{V}_k} \lceil \frac{r|\mathcal{F}_i|}{1-r} \rceil \geq |\mathcal{V}_{k+1}|, \forall k \in [M-1, K-1] \end{aligned} \tag{1}$$

We have two remarks. First, $\alpha \in [0, 1]$ is a adjustable weight that reflects the relative importance of coverage and delay, and β is a fixed scaling factor to unify two different objective units. Second, the last constraint is due to the aforementioned third design constraint.

The above optimization problem can be viewed as a vari-

ation of classical set partition problem (SPP), which is NP-hard. In what follows, we introduce a heuristic approximation solution by constructing a collection of disjointed subsets $\{\mathcal{V}_k\}_{k=0}^{K}$ from the botnet set.

4) **Heuristic Solution***:* Our intuition is to use all the budget c and fill the first M hops of the retweeting trees with the bots having the lowest suspension cost in terms of the number of lost non-bot followers. We then recursively place the bots with the highest number of non-bot followers from level $M+1$ to the last level, in order to reduce the average latency as well as cost.

To begin with, our approximation is built on the solutions to the traditional maximum coverage problem (or MAXCOVER) and the minimum coverage problem (MINCOVER), which is to select k sets from a collection of subsets of a ground set so that the their union is maximized [31] or minimized, respectively. MAXCOVER and MINCOVER problems are both NP-hard and have greedy approximation algorithms by iteratively selecting the maximum or minimum subset after extracting the selected elements, respectively.

Our solution consists of the following two steps. First, given the budget c for $\mathcal{S}$ and that all the followers in $\tilde{\mathcal{S}}$ will be lost because of the suspension, we minimize the objective $|\tilde{\mathcal{S}}|$ by using MINCOVER to choose c bots as $\tilde{\mathcal{S}}$ with the minimum total number of non-bot followers. In doing so, we can determine the union of the bot set for the first M level. The bot subset in each level will be determined later. Here we just assume that $\tilde{\mathcal{S}}$ has been divided into the M groups, each corresponding to the bots in one of the first M levels. Second, we construct $\{\mathcal{V}_k\}_{k=M+1}^{K}$ to greedily increase the coverage C and at the same time lower the average delay T. Specifically, assuming that we have known $\mathcal{V}_M$, to determine $\mathcal{V}_{M+1}$, we first set the cardinality of $\mathcal{V}_{M+1}$ be equal to $\sum_{i\in\mathcal{V}_k}\lceil\frac{r|\mathcal{F}_i|}{1-r}\rceil$ according to the last constraint in (1) and then use MAXCOVER to choose a subset of $|\mathcal{V}_{M+1}|$ bots from the remaining bot set with the maximum number of non-bot followers. We repeat this greedy selection for every level $k = M+2,\ldots,K$.

The remaining problem is how to partition the $\tilde{\mathcal{S}}$ into M subsets, each corresponding to one of the first M levels. A heuristic observation here is that we need to maximize $|\mathcal{V}_M|$, as the more non-bot followers of the social bots in the Mth level, the more social bots in level $M+1$ and subsequent levels, and also the lower average delay according to the last constraint in (1). Given the budget $|\tilde{\mathcal{S}}| = c$, we obtain $|\mathcal{V}_M|_{max} = c-M+1$ when the retweeting forest has a single tree whose first $M-1$ levels form a straight line, as shown in Fig. 1(b). The bots in the Mth level is then determined by using MAXCOVER to choose the $c-M+1$ bots from $\tilde{\mathcal{S}}$ with the maximum number of non-bot followers.

To determine the level for each of the remaining $M-1$ bots, we sort the remaining $M-1$ bots in $\tilde{\mathcal{S}}$ in the descending order according to the number of their non-bot followers and assign them to the corresponding level, e.g., the bot with the highest number of non-bot followers will be assigned to the first level. Note that it is possible that after we maximizing the number of bots at the Mth level, the remaining bots are less than the allowed on the Mth level, so the $(M+1)$-th is not full. To further reduce the average delay in such cases, we move the 'unused' bots in the Mth level to the first level.

After determining $\{\mathcal{V}_i\}_{i=1}^{K}$ from the social-bot set $\{1,\ldots,n\}$, we can then build the final retweeting forest (tree). Specifically, the number of retweeting trees is equal to the cardinality of $\mathcal{V}_1$, which is one if the $(M+1)$-th level is full or greater than one otherwise. We then randomly choose one social bot from $\mathcal{V}_1$ to be the root of the retweeting tree with more than one level, which is followed by the bots from the second to Mth level determined by $\mathcal{V}_2$ to $\mathcal{V}_M$, respectively. Finally, we build the level from $k = M+1$ to K by selecting certain number of social bots from $\mathcal{V}_k$ according the last constraint in Eq. (1).

C. Trace-driven Evaluation

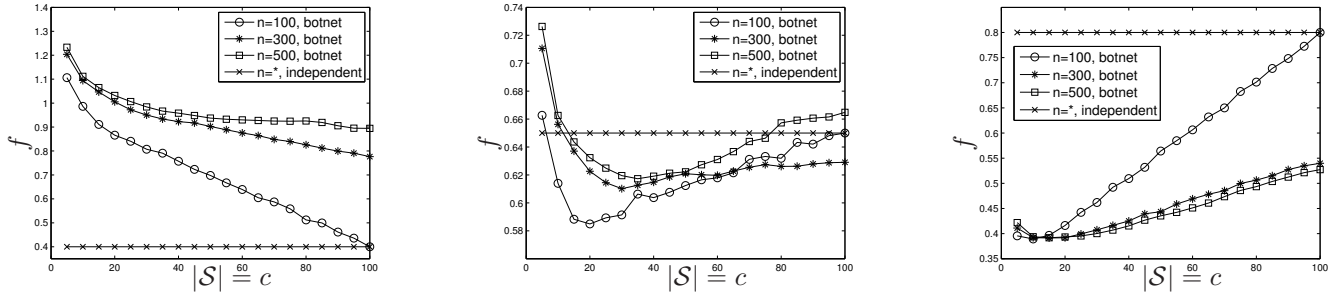
We conduct trace-driven simulations to compare the performance of spam distribution using the independent and botnet methods, as well as evaluating the tradeoffs among the multiple goals in the botnet method.

The evaluation for independent bots is straightforward. In particular, given the bot set $\mathcal{V}$ with $|\mathcal{V}| = n$, we place all the bots in the first level which will be suspended completely. We then have $\mathcal{C} = \tilde{\mathcal{S}} = n$, and $\tau = 0$. The single objective in Problem (1) is thus $f = \alpha$.

To evaluate the botnet method, we set up the simulations according to existing measurement data and heuristics. We set $K = 10$, and $t_1 = 0, t_i = 0.5i$ hour for $i = 2,\ldots K$ according to [24]. To build $\{\mathcal{F}_i\}_{i=1}^{n}$, we generate $|\mathcal{F}_i|$ according to the Gaussian distribution with $\mu = 32$ as the average number of followers in the dataset of [24]. We also set the variance to $\sigma^2 = 5$, generate a legitimate follower set $\mathcal{F}$ with $|\mathcal{F}| = 6000$,[4] and randomly choose $|\mathcal{F}_i|$ followers from the set $\mathcal{F}$ for each bot i. In addition, according to [10], the average path length of the spammer community is 2.60, so we set $M = 3$ to suspend the bots in the first three hops of $\mathcal{F}$. Finally, we set β to one and the retweeting ratio $r = 0.2$. Due to space constraints, we focus on on the impact of α, c, and n and do not report the impact of $|\mathcal{F}|$, σ^2, r, M, or β in this paper.

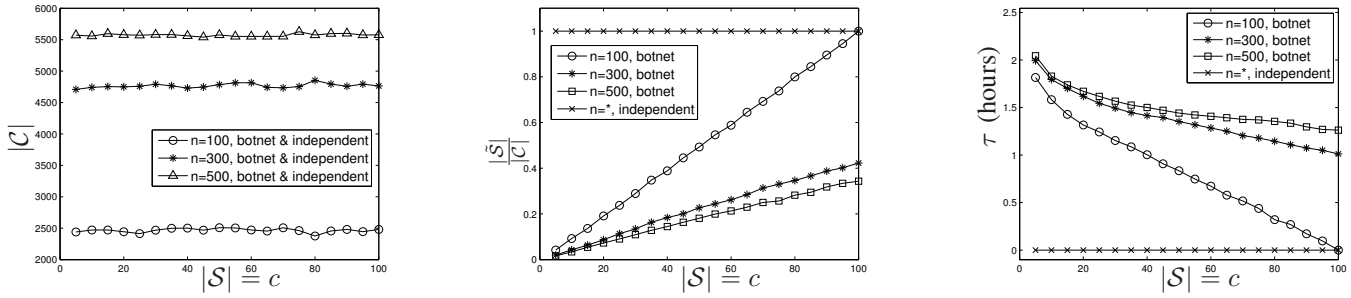
Fig. 2 compares the independent and botnet methods using the objective function f with different weights α. As stated before, f is simply equal to α for the independent case because $\tau = 0$ and $\tilde{\mathcal{S}} = \mathcal{C}$. For the botnet method, the objective f is the weighted sum of $\frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|}$ and the delay τ. When α is small, τ has higher impact on f than $\frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|}$, while when α is large, $\frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|}$ will dominate f. Specifically, we can see from Fig. 2(a) that when $\alpha = 0.4$, the independent method outperforms the botnet method with smaller f. However, as shown in Fig. 2(b), when α increases to 0.65, the botnet method can achieve lower f than the independent method does. This trend is more obvious in Fig. 2(c) where $\alpha = 0.8$ for the same reason.

[4]For the Gaussian distribution with $\mu = 32$ and $\sigma^2 = 5$, the probability for generating a negative $|\mathcal{F}_i|$ is negligible.



(a) $\alpha = 0.4$ (b) $\alpha = 0.65$ (c) $\alpha = 0.8$

Fig. 2. Performance comparison of independent and botnet method in spam distribution at different αs in terms of the single objective f.



(a) Coverage $|\mathcal{C}|$ (b) The number of lost legitimate followers $\frac{|\tilde{\mathcal{S}}|}{C}$ (c) Average delay τ (hours)

Fig. 3. Performance comparison of independent and botnet method in spam distribution in terms of separate objective.

In addition, Fig. 2 demonstrates the tradeoff between the objective f and the budget $|\mathcal{S}| = c$ with different weights α for the botnet method. In particular, we can see from Figs. 2(a) to 2(c) that there is no global optimal point where f and budget c are simultaneously minimized. Instead, for given pair of α and n (i.e., the number of social bots), we can find a Pareto optimal curve on which f and c cannot be further reduced at the same time.

Moreover, we can see from Fig. 2 that when the budget c is close to the number of bots, the independent method and the botnet method can achieve the similar objective f. This is expected because most bots are allowed to be suspended given the large budget, and the solution output by our heuristic is similar to that of independent method. For example, when $n = c$, we can place all bots in the first level to minimize the average delay.

Figs. 2(a) to 2(c) compare the two methods in terms of coverage, the number of lost legitimate followers, and the average delay. We can see that both methods have the same coverage $|\mathcal{C}|$, which is equal to $|\mathcal{F}|$, as well as the maximum value of C. In addition, we can see from Fig. 3(c) that the delay of the independent method is zero, while that of botnet method could be on the order of hours. Finally, Fig. 3(b) shows that the botnet method has significant advantage than the independent method in terms of $|\tilde{\mathcal{S}}|$, the number of lost legitimate followers, as $|\tilde{\mathcal{S}}|$ is always equal to $|\mathcal{F}|$ for the independent scheme.

Fig. 3 shows various relations between separate objectives $|\mathcal{C}|$, $\frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|}$, τ and $|\mathcal{S}|$ for the botnet method. In particular, Fig. 3(a) depicts that the botnet method maintains a constant coverage $|\mathcal{C}|$ for the given budget c at different n. Actually, we further verified that this value is the maximum coverage under a specific n. This is expected since $K = 10$ is large enough to include all of n bots in the retweeting tree. In addition, it is also obvious that the cost $|\tilde{\mathcal{S}}|$ linearly increases with the budget $|\mathcal{S}|$ as shown in Fig. 3(b) and that the delay decreases as the increasing of budget in Fig. 3(c).

Finally, the botnet size n also has some impact on separate objectives in the botnet case. Intuitively, the larger n, the larger coverage as shown in Fig. 3(a). In addition, Fig. 3(b) shows that $\frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|}$ decreases as n increases. The reason is that the larger n, the more bots with less non-bot followers will be assigned to the first M levels, resulting in smaller $|\tilde{\mathcal{S}}|$ and thus larger $\frac{|\tilde{\mathcal{S}}|}{|\mathcal{C}|}$. In addition, Fig. 3(c) shows that the larger n, the higher the average delay τ, which is also expected.

In summary, from the view point of the botmaster, these evaluations show that the botnet scheme is more flexible than independent method when considering multiple objectives of the spam distribution at the same time.

IV. Social Botnet for Digital-influence Manipulation

In this section, we start with a brief introduction to digital influence and then experimentally show the efficacy of using the social botnet to manipulate digital influence.

A. Rise of Digital Influence

Digital influence is one of the hottest trends in social media and is defined as "the ability to cause effect, change

behavior, and drive measurable outcomes online" in [32]. The huge commercial potential of digital influence is in line with the increasingly recognized importance of word-of-mouth marketing on social networks. There are also growing business cases in which various companies successfully promoted their services/products by reaching out to most influential social-network users in their respective context [32].

The future of digital influence also relies on effective tools to measure it. As reported in [32], there are over 20 popular digital-influence software vendors such as Klout, Kred, and Retweet Rank. Every vendor has its proprietary method to compute an *influence score* for a given user based on his activities within his affiliated social network such as Twitter, Facebook, Google+, and LinkedIn, and higher scores represent greater influence. Moreover, the typical business model of digital-influence vendors is based around connecting businesses with individuals of high influence. Companies have paid to get in contact with individuals with high influence scores in hopes that free merchandise and other perks will influence them to spread positive publicity for them. For example, Klout announced a growth of 2000 new partners over a one year period in May 2012.

B. Botnet-based Digital-influence Manipulation

Given the great potential of digital influence, whether it can be maliciously manipulated is an important research issue. For example, assume that malicious users could collude to significantly increase their influence scores. A company using the digital-influence service may consider them most influential and choose them as the targets of important marketing campaigns by mistake, thus having potentially huge financial loss, while malicious users can potentially benefit, e.g., by getting free sample products. In addition, malicious users may attract more legitimate followers who tend to follow most influential users and thus become more influential.

As the first work of its kind, we now explore the feasibility of using the botnet to manipulate digital influence. Our studies involve three most popular digital-influence vendors for Twitter users: Klout, Kred, and Retweet Rank. For clarity, we summarize their key features as follows.

- Klout: The Klout score of a Twitter user is on the scale of 1 to 100 and updated daily based on how frequently he or she is retweeted and mentioned in the last 90 days.[5] The average Klout score is close to 20, and the score of the 80th percentile is over 40.
- Kred: The Kred score of a Twitter user is on the scale of 1 to 1,000 and updated in real time according to how frequently he or she is retweeted, replied, mentioned, and followed on Twitter in the last 1,000 days.[6]
- Retweet Rank: It ranks the users based on how many times they each have been retweeted recently and how many followers/friends they each have.[7] Retweet ranks are updated on an hourly basis, and a retweet rank of x means that the corresponding user is the xth most influential on Twitter. A retweet rank can also be translated into a percentile score ranging from 1 to 100, indicating how the user score relative to other Twitter users.

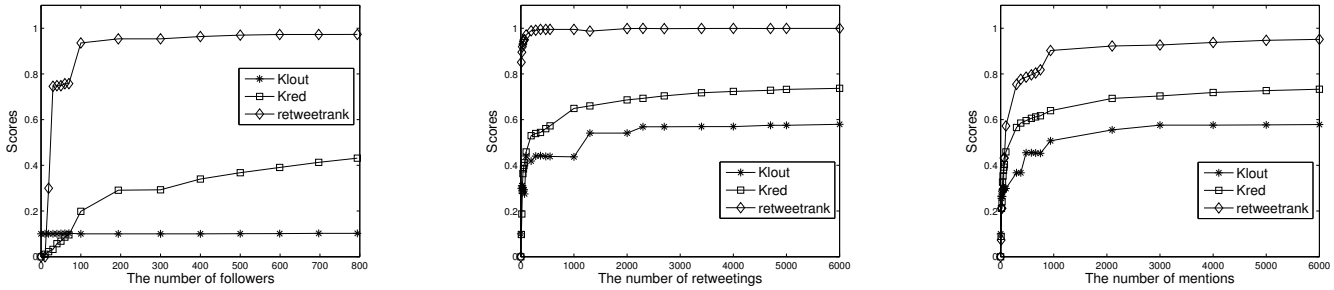
Given a social botnet of n bots, we want to investigate whether it is feasible to generate an arbitrary influence score d_i for every bot $i \in [1, n]$ under each of the above three tools. Since every bot is usually indistinguishable from a legitimate user, our investigation can also shed light on the feasibility of using the botnet to manipulate the influence score of an arbitrary Twitter user. Since every digital-influence vendor (including the above three) usually keeps confidential its detailed algorithm for computing influence scores, our studies are purely based on real Twitter experiments. Due to tight space constraints, we present and discuss two different experiments in what follows.

1) **Impact of Different Actions on Twitter***:* Since almost all digital-influence tools measure the influence of a Twitter user as his ability to drive others to actions, our first experiment aims to evaluate the impact of different actions on Twitter, including following, retweeting, and mentioning. Note that we do not evaluate the impact of replying which is expected to have the same effect as mentioning, as replies are also considered mentions on Twitter.

The first experiment involves $n = 1,000$ bots, all of which have not been scored by Klout, Kred, or Retweet Rank. Note that Klout assigns an influence score of 10 to new users, while Kred and Retweet Rank both assign a zero score to new users. We randomly choose three disjoint groups of bots, each containing 10 bots and for a unique action. For every bot in the *following* group, we add 10 followers every day during the first 10 days and then 100 followers every day during the next 10 days which are randomly selected from the botnet. Likewise, every bot in the *retweeting* (or *mentioning*) group is retweeted (or mentioned) by random bots in the botnet 10, 100 and 1000 times every day during the first, second, and the last 10 days, respectively. Since we averaged the actions on all the bots, the daily actions for each bot is too trivial to be suspected by Twitter. Given the different score-updating schedules of the three vendors, we choose to report the influence score of every bot observed at every midnight. In addition, since the three vendors have different score scales, every influence score is normalized with respect to the corresponding maximum score to facilitate the comparison. In particular, we show $x/100$ and $y/1000$ instead for a Klout score x and a Kred score y, respectively, and adopts the percentile score for Tweet Rank.

Figs. 4(a)∼4(c) show the impact of following, retweeting, and mentioning actions on Klout, Kred, and Retweet Rank influence scores, where every data point is the average across the same bot group. We can see from Fig. 4(a) that the Klout influence score is not affected by the number of followers, while both Kred and Retweet Rank influence scores increase as the number of followers increases. This indicates that the members of a botnet can easily boost their Kred and Retweet Rank influence scores by purposefully following each other.

[5]http://corp.klout.com/blog/category/understanding-the-klout-score/
[6]http://kred.com/rules
[7]http://www.retweetrank.com/view/about



(a) Impact of following (b) Impact of retweeting (c) Impact of mentioning

Fig. 4. Manipulating digital influence by following, retweeting, and mentioning.

In addition, we can see from Fig. 4(b) that all three types of influence scores increase as to the number of retweets resulted from the user. On the one hand, this makes much sense, since the higher the frequency of a user being retweeted, the higher influence of that user has in its local neighborhood. On the other hand, this also leaves the influence score system vulnerable to botnets, in which collaborating bots can fake arbitrarily high retweeting frequency for any user. Similar trends can be seen in Fig. 4(c) for the impact of mentioning. Note that each curve in Figs. 4(b) and 4(c) shows the difficulty of escalating the influence score to an extremely high position, which results from the score setting of the vendors [33]–[35].

2) **Speed of Digital-influence Manipulation***:* Since our first experiment have shown that retweeting is the most effective way in manipulating Klout, Kred, and Retweet Rank scores, we now focus on how fast retweeting can affect the influence scores using the same botnet in the second experiment. For this purpose, we randomly select another three groups of bots from the botnet, each containing 10 bots different from the ones in the first experiment. Every bot in the first, second, and third groups is retweeted 10, 100, and 1,000 times every day by random bots until the scores reach the 80th percentile, which corresponds to 40, 600, and 80 in Klout, Kred, and Retweet Rank, respectively.

Fig. 5 further shows the impact of retweeting frequency on the change of influence scores, where every data point represents the average across the same bot group. In particular, we can see that the number of days needed to increase the group average influence score from initial value to 80% percentile is approximately inversely proportional to the retweeting frequency. In addition, we can see that for all three vendors, it is possible to reach the 80% percentile with only single day by retweeting 1000 times per day.

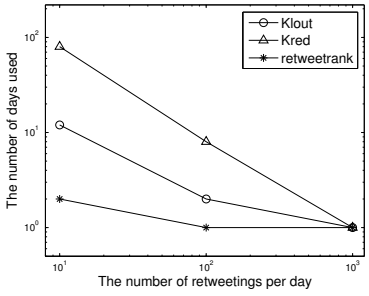


Fig. 5. Under different speed of retweeting, the number of days need to manipulate digital influence scores from nothing into 80-th percentile.

V. Related Work

Due to space constraints, we only discuss the prior work most germane to our work in this paper.

Boshmaf *et al.* showed that a social botnet is very effective in connecting to many random or targeted Facebook users (i.e., large-scale infiltration) [2]. In contrast, our work has a different purpose and investigates the efficacy of using the social botnet for spam distribution and digital-influence manipulation.

There is a rich literature on spam detection in OSNs. The first line of work such as [8], [21], [25]–[30], [36], [37] considers independent spam bots and comes up with different methods to characterize and detect them. Some of these results such as [27] have been incorporated into our design constraints in §III-B2. The second line of work such as [3]–[5], [10], [11], [20] focuses on characterizing and detecting organized spam campaigns launched by an army of spam bots. We actually discover a new method for effective spam campaign in this paper, and whether the results in [3]–[5], [10], [11], [20] can be directly or adapted to detect our method is certainly challenging and worthy of further investigation.

Finally, there are effective solutions such as [38], [39] to detecting Sybil accounts in distributed systems under the control of a single attacker. These solutions commonly assume that the link between two accounts corresponds to a real social relationship difficult to establish and impossible to forge. This assumption, however, does not hold in Twitter which allows followings without prior consent. Therefore, these solutions is ineffective in detecting the social botnet in our case.

VI. Conclusion and Future Work (Possible Countermeasures)

In this paper, we firmly validated the efficacy and merits of botnet-based spam distribution and digital-influence manipulation on Twitter through thorough experiments and trace-driven simulations. As the future work, we will first extend our studies to other OSNs such as Facebook and Google+.

We will also investigate other attacks that can be enabled or facilitated by the social botnet so as to raise the alertness of OSN users and also help OSNs improve their misbehavior-detection systems. In addition, we plan to investigate three lines of countermeasures against our attacks.

The first line is motivated by the observation that the amount of interactions from a legitimate OSN user to a social bot is usually far less than that in the reverse direction. We thus seek to combine this observation with existing Sybil defenses [38], [39] to effectively identify large-scale social botnets.

Another possible defense is to identify malicious applications registered by the botmaster at OSNs. In particular, a large-scale social botnet often involves delegating the access privileges of individual bots to the applications the botmaster develops based on the OSN's open APIs and registers with the OSN, as described in §II. The botmaster can then simultaneously control the bots via the applications. Since the OSN like Twitter normally rate-limits the number of open API calls made by a single application (e.g., 350 calls per hour in Twitter), the botmaster will have to register and simultaneously operate many applications for a large-scale social botnet. In contrast, a legitimate user may mostly use the conventional web interface and occasionally rely on third-party applications to access the OSN. Moreover, a malicious application has little chance to attract legitimate users who are unlikely to delegate their access privileges to unknown third-party applications. These observations can help design effective and efficient algorithms for OSNs to identify malicious botnet applications.

Finally, our experiments for digital-influence manipulation reveal that digital-influence vendors barely consider the existing influence of OSN users in deriving new influence scores. Intuitively, the interactions with influential users should be counted more than those with non-influential users. Based on this observation, we plan to investigate new digital-influence quantization methods immune to botnet-based manipulation.

REFERENCES

[1] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All your contacts are belong to us: automated identity theft attacks on social networks," in *WWW'09*, Madrid, Spain, 2009, pp. 551–560.

[2] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu, "The socialbot network: when bots socialize for fame and money," in *ACSAC'11*, Orlando, Florida, 2011, pp. 93–102.

[3] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Zhao, "Detecting and characterizing social spam campaigns," in *IMC'10*, 2010, pp. 35–47.

[4] C. Grier, K. Thomas, V. Paxson, and M. Zhang, "@spam: the underground on 140 characters or less," in *CCS'10*, 2010, pp. 27–37.

[5] K. Thomas, C. Grier, V. Paxson, and D. Song, "Suspended accounts in retrospect:an analysis of twitter spam," in *IMC'11*, Berlin, Germany, Nov. 2011.

[6] J. Ratkiewicz, M. Conover, M. Meiss, B. Gonçalves, A. Flammini, and F. Menczer, "Detecting and tracking political abuse in social media," in *ICWSM'11*, 2011.

[7] J. Ratkiewicz, M. Conover, M. Meiss, B. Gonçalves, S. Patil, A. Flammini, and F. Menczer, "Truthy: mapping the spread of astroturf in microblog streams," in *WWW '11*, 2011, pp. 249–252.

[8] A. H. Wang, "Don't follow me - spam detection in twitter," in *SECRYPT'10*, Jul. 2010, pp. 142–151.

[9] S. Ghosh, G. Korlam, and N. Ganguly, "Spammers' networks within online social networks: a case-study on twitter," in *WWW'11*, 2011, pp. 41–42.

[10] 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 *WWW'12*, Apr. 2012.

[11] S. Ghosh, B. Viswanath, F. Kooti, N. K. Sharma, K. Gautam, F. Benevenuto, N. Ganguly, and K. Gummadi, "Understanding and combating link farming in the twitter social network," in *WWW'12*, Lyon, France, Apr. 2012.

[12] "The twitter rules." [Online]. Available: https://support.twitter.com/articles/18311-the-twitter-rules

[13] M. Cha, H. Haddadi, F. Benevenuto, and K. Gummadi, "Measuring user influence in twitter: The million follower fallacy," in *ICWSM'10*, May 2010.

[14] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "TwitterRank: finding topic-sensitive influential twitterers," in *WSDM'10*, New York, NY, USA, 2010, pp. 261–270.

[15] S. Williams, "Chevy gives 3-day sonic drives to those with big klout," Nov. 2011. [Online]. Available: http://adage.com/article/news/chevy-tests-sonics-high-klout-scores/231220/

[16] "Klout perks: A healthy two years." [Online]. Available: http://corp.klout.com/blog/2012/06/klout-perks-a-healthy-two-years/

[17] A. Knapp, "A high klout score can lead to better customer service," Jun. 2012. [Online]. Available: http://www.forbes.com/sites/alexknapp/2012/06/12/a-high-klout-score-can-lead-to-better-customer-service/

[18] V. Hernandez, "Measuring influence online: A q&a with klout's ceo," Feb. 2012. [Online]. Available: http://www.cnn.com/2012/02/23/tech/social-media/klout-joe-fernandez/index.html?hpt=hp_bn6

[19] "Oauth open authentication system." [Online]. Available: http://oauth.net/

[20] Z. Chu, I. Widjaja, and H. Wang, "Detecting social spam campaigns on twitter," in *ACNS'12*, Jun. 2012.

[21] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time url spam filtering service," in *IEEE S&P'11*, 2011, pp. 447–462.

[22] R. Mac, "Twitter acquires web security firm dasient," Jan. 2012. [Online]. Available: http://www.forbes.com/sites/ryanmac/2012/01/24/twitter-acquires-web-security-firm-dasient/

[23] Z. Coburn and G. Marra, "Realboy: Believable twitter bots," 2008. [Online]. Available: http://ca.olin.edu/2008/realboy/index.html

[24] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *WWW'10*, Raleigh, North Carolina, USA, 2010, pp. 591–600.

[25] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: social honeypots + machine learning," in *SIGIR'10*, 2010, pp. 435–442.

[26] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *ACSAC'10*, 2010.

[27] J. Song, S. Lee, and J. Kim, "Spam filtering in twitter using sender-receiver relationship," in *RAID'11*, Sep. 2011.

[28] C. Yang, R. Harkreader, and G. Gu, "Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers," in *RAID'11*, Sep. 2011.

[29] C. Zhang and V. Paxson, "Detecting and analyzing automated activity on twitter," in *PAM'11*, Mar. 2011, pp. 102–111.

[30] H. Gao, Y. Chen, K. Lee, D. Palsetia, and A. Choudhary, "Towards online spam filtering in social networks," in *NDSS'12*, San Diego, CA, Feb. 2012.

[31] F. Chierichetti, R. Kumar, and A. Tomkins, "Max-cover in map-reduce," in *WWW'10*, 2010.

[32] B. Solis, "The rise of digital influence," Altimeter Group, Research Report, Mar. 2012.

[33] "Klout." [Online]. Available: http://klout.com/

[34] "Influence in kred." [Online]. Available: http://kred.com/rules

[35] "Faq about retweet rank." [Online]. Available: http://www.retweetrank.com/view/about

[36] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammers on twitter," in *CEAS'10*, Jul. 2010.

[37] S. Lee and J. Kim, "WARNINGBIRD: Detecting suspicious urls in twitter stream," in *NDSS'12*, San Diego, CA, Feb. 2012.

[38] H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman, "Sybilguard: defending against sybil attacks via social networks," in *SIGCOMM'06*, Pisa, Italy, 2006, pp. 267–278.

[39] B. Viswanath, M. Mondal, A. Clement, P. Druschel, K. Gummadi, A. Mislove, and A. Post, "Exploring the design space of social network-based sybil defenses," in *COMSNETS'12*, Bangalore, India, Jan. 2012.