The Socialbot Network: When Bots Socialize for Fame and Money

Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, Matei Ripeanu
University of British Columbia
Vancouver, Canada
{boshmaf,ildarm,beznosov,matei}@ece.ubc.ca

ABSTRACT

Online Social Networks (OSNs) have become an integral part of today's Web. Politicians, celebrities, revolutionists, and others use OSNs as a podium to deliver their message to millions of active web users. Unfortunately, in the wrong hands, OSNs can be used to run astroturf campaigns to spread misinformation and propaganda. Such campaigns usually start off by infiltrating a targeted OSN on a large scale. In this paper, we evaluate how vulnerable OSNs are to a large-scale infiltration by *socialbots*: computer programs that control OSN accounts and mimic real users. We adopt a traditional web-based botnet design and built a *Socialbot Network* (SbN): a group of adaptive socialbots that are orchestrated in a command-and-control fashion. We operated such an SbN on Facebook—a 750 million user OSN—for about 8 weeks. We collected data related to users' behavior in response to a large-scale infiltration where socialbots were used to connect to a large number of Facebook users. Our results show that (1) OSNs, such as Facebook, can be infiltrated with a success rate of up to 80%, (2) depending on users' privacy settings, a successful infiltration can result in privacy breaches where even more users' data are exposed when compared to a purely public access, and (3) in practice, OSN security defenses, such as the Facebook Immune System, are not effective enough in detecting or stopping a large-scale infiltration as it occurs.

1. INTRODUCTION

Online Social Networks (OSNs) such as Facebook[1] and Twitter[2] have far exceeded the traditional networking service of connecting people together. With millions of users actively using their platforms, OSNs have attracted third parties who exploit them as an effective media to reach and potentially influence a large and diverse population of web users [21, 23]. For example, during the 2008 U.S. presidential election, social media was heavily employed by Obama's campaign team who raised about half a billion dollars online, introducing a new digital era in presidential fundraising [40]. Moreover, it has been argued that OSNs, as democracy-enforcing communication platforms, were one of the key enablers of the recent Arab Spring in the Middle East [35, 38]. Such a global integration of social media into everyday life is rapidly becoming the norm, and arguably is here to stay [8]. But what if some of the content in social media—OSNs in particular—is not written by human beings?

A new breed of computer programs called *socialbots* are now online, and they can be used to influence OSN users [24]. A socialbot is an automation software that controls an account on a particular OSN, and has the ability to perform basic activities such as posting a message and sending a connection request. What makes a socialbot different from self-declared bots (e.g., Twitter bots that post up-to-date weather forecasts) and spambots is that it is designed to be stealthy, that is, it is able to pass itself off as a human being. This allows the socialbot to compromise the social graph of a targeted OSN by infiltrating (i.e., connecting to) its users so as to reach an influential position. This position can be then exploited to spread misinformation and propaganda in order to bias the public opinion [26]. For example, Ratkiewicz et al. [33] describe the use of Twitter bots to run astroturf and smear campaigns during the 2010 U.S. midterm elections.

As socialbots infiltrate a targeted OSN, they can further harvest private users' data such as email addresses, phone numbers, and other personal data that have monetary value. To an adversary, such data are valuable and can be used for online profiling and large-scale email spam and phishing campaigns [30]. It is thus not surprising that different kinds of socialbots are being offered for sale in the Internet black-market for as much as $29 per bot [4].

Recently, many techniques have been proposed that aim to automatically identify spambots in OSNs based on their abnormal behavior [31, 16, 37]. For example, Stein et al. [36] present the Facebook Immune System (FIS): an adversarial learning system that performs real-time checks and classification on every read and write action on Facebook's database, all for the purpose of protecting its users and the social graph from malicious activities. It is, however, not well-understood how such defenses stand against socialbots that mimic real users, and what the expected users' behavior might be in response to a large-scale infiltration by such bots.

In this paper, we aim to fill this knowledge gap. We treat large-scale infiltration in OSNs as an organized campaign that is run by an army of socialbots to connect to either random or targeted OSN users on a large scale. Therefore,

[1]`http://www.facebook.com`
[2]`http://www.twitter.com`

we decided to adopt a traditional web-based botnet design and define what we call a *Socialbot Network* (SbN): a group of re-programmable socialbots that are orchestrated by an adversary (called a *botherder*) using a software controller (called a *botmaster*). The botmaster is designed to exploit the known properties of social networks, such as the *triadic closure principle* [32], and use them as heuristics to define commands, which increase the magnitude of the potential infiltration in the targeted OSN.

We built a simple, yet effective, SbN consisting of 102 socialbots and a single botmaster. We then operated this SbN on Facebook for 8 weeks. During that time, the socialbots were able to send a total of 8,570 connection requests. We recorded all data related to the anticipated infiltration and the corresponding users' behavior, along with all accessible users' profile information. Overall, we summarize our main findings as follows:

(1) OSNs, such as Facebook, are highly vulnerable to a large-scale infiltration. From the OSN side, we show that it is not difficult to fully automate the overall operation of an SbN, including accounts creation. From the users' side, we show that most OSN users are not careful enough when accepting connection requests sent by strangers, especially when they have mutual connections. This behavior can be exploited to achieve a large-scale infiltration with a success rate of up to 80%.

(2) Depending on users' privacy settings, operating an SbN can result in many privacy breaches. We show that greater number of users' data can be harvested after a large-scale infiltration. This data include email addresses, phone numbers, and other profile information, all of which have monetary value. Unfortunately, this also includes the private data of users who have not been infiltrated, but are connected to infiltrated users. Moreover, we show that a botherder can operate an SbN conservatively, at a slow pace, and still collect an average of 175 new chunks of publicly-unaccessible users' data per socialbot per day.

(3) In practice, OSN security defenses such as the FIS are not effective enough in detecting a large-scale infiltration. Our results show that a successful infiltration of an OSN user is expected to be observed within the first 3 days after the request has been sent by a socialbot. This means that the social graph will rapidly change in a relatively short time, and the socialbots will get gradually integrated into the targeted online community. We found that the FIS was able to block only 20% of the accounts used by the socialbots. This, however, was a result of feedback from users who flagged these accounts as spam. In fact, we did not observe any evidence that the FIS detected what was really going on: an organized large-scale infiltration campaign.

The rest of the paper is organized as follows: We first provide background information and define our notations in Section 2. After that, we present the concept of a Socialbot Network, along with its design goals and construction details, in Section 3. Next, we demonstrate our experiments with an SbN on Facebook in Section 4, and then we discuss our results in Section 5. This is followed by an outline of related works in Section 6. Finally, we conclude the paper in Section 7.

2. PRELIMINARIES

In what follows, we present background information and define the notations we use in the upcoming discussion.

2.1 Online Social Networks

Online Social Networks (OSNs) provide centralized web platforms that facilitate users' social activities. A user in such a platform owns an account and is represented by a profile that describes her social attributes such as name, gender, interests and contact information. We use the terms "account", "profile", and "user" interchangeably. A social connection between two users can be either undirected like friendships in Facebook, or directed like follower-followee relationships in Twitter.

An OSN can be modeled as a graph $G = (V, E)$, where V represents a set of users and E represents a set of social connections among these users. For every user $u \in V$, the set $\Gamma(u)$ is called the *neighborhood* of u, and it contains all users in V with whom u has social connections. We denote the average neighborhood size in G by $N_{\texttt{avg}} = |V|^{-1} \sum_{u \in V} |\Gamma(u)|$. Finally, we call the set $\Delta(u) = \bigcup_{v \in \Gamma(u)} \Gamma(v)$ the *extended neighborhood* of u.

2.2 Social Engineering and Socialbots

Traditionally, *social engineering* is defined as the art of gaining access to secure objects by exploiting human psychology, rather than using hacking techniques. Social engineering, however, has become more technical and complex; social engineering attacks are being computerized and fully automated, and are becoming adaptive and context-aware [9, 5]. In fact, some of these attacks are sophisticated and use learned or hard-coded heuristics and observations about users' behaviour in the targeted system so as to increase the magnitude of their potential damage [5, 6, 20].

The next generation of social engineering attacks is even more deceptive; they employ an automation software called a *socialbot* that controls a profile in an OSN, and has the ability to execute basic online social activities. For example, Realboy [10] is an experimental project that aims to design believable Twitter bots that imitate real Twitter users.

2.3 OSN Vulnerabilities

We discuss four vulnerabilities found in today's OSN which allow an adversary to carry out a large-scale infiltration campaign. We treat each vulnerability separately and provide evidence to support it.

2.3.1 Ineffective CAPTCHAs

OSNs employ CAPTCHAs [42] to prevent automated bots from abusing their platforms. An adversary, however, can often circumvent this countermeasure by using different techniques such as automated analysis using optical character recognition [6], exploiting botnets to trick infected victims into manually solving CAPTCHAs [5, 12], reusing session IDs of known CAPTCHAs [18], cracking MD5 hashes of CAPTCHAs that are validated at the client side [44], and hiring cheap human labor [27].

Let us consider the use of cheap human labor to break CAPTCHAs; a phenomenon that is known as CAPTCHA-breaking business. Motoyama et al. [27] show that companies involved in such a business are surprisingly efficient; they have high service quality with a success rate of up to 98%, charge $1 per 1,000 successfully-broken CAPTCHAs, and provide software APIs to automate the whole process. Thus, even the most sophisticated CAPTCHA technology that only humans could solve can be effectively circumvented with a small investment from an adversary. In such a situation, the adversary acts as an economist; he would invest in such businesses if the return on investment is considerably high. This allows researchers to look at online attacks from an economic context, and define cost metrics that measure when it is economically feasible for an adversary to mount a large-scale attack that involves, for instance, breaking CAPTCHAs through employing cheap human labor [17].

2.3.2 Fake User Accounts and Profiles

Creating a user account on an OSN involves three tasks: providing an active email address, creating a user profile, and sometimes solving a CAPTCHA. Each user account maps to one profile, but many user accounts can be owned by the same person or organization using different email addresses. In what follows, we argue that an adversary can fully automate the account creation process. This, however, is not new, as similar tools are used for online marketing [2].

Fake user accounts: When creating a new user account in an OSN, an email address is required to first validate and then activate the account. The OSN validates the account by associating it to the owner of the email address. After account validation, its owner activates the account by following an activation link that is emailed by the OSN. Accordingly, an adversary has to overcome two hurdles when creating a new account: providing an active email address that he owns, and account activation. To tackle the first hurdle, the adversary can maintain many emails by either using "temp" email addresses that are obtained from providers that do not require registration such as `10minutemail.com`, or by creating email addresses using email providers that do not limit the number of created emails per browsing session or IP address such as `mail.ru`. As for the second hurdle, an adversary can write a simple script that downloads the activation email, and then sends an HTTP request to the activation URL that is included in the downloaded email.

Fake user profiles: Creating a user profile is a straightforward task for real users; they just have to provide the information that represents their social attributes. For an adversary, however, the situation is a bit different. The objective of the adversary is to create profiles that are "socially attractive". We consider a purely adversarial standpoint concerning social attractiveness; the adversary aims to exploit certain social attributes that have shown to be effective in getting users' attention. Such attributes can be inferred from recent social engineering attacks. Specifically, using a profile picture of a good looking woman or man has had the greatest impact [6, 14]. Thus, an adversary can use publicly available personal pictures for the newly created profiles, with the corresponding gender and age-range. In fact, the adversary can use already-rated personal pictures from websites like `hotornot.com`, where users publicly post their personal pictures for others to rate their "hotness".[3] It is thus possible for an adversary to automate the collection of the required profile information through crawling (or scavenging in this case) the Web.

[3]Such sites also provide categorization of the rated personal pictures based on gender and age-range.

2.3.3 Crawlable Social Graphs

The social graph of an OSN is usually hidden from public access in order to protect its users' privacy. An adversary, however, can reconstruct parts of the social graph by first logging in to the OSN platform using an account, and then traversing through linked user profiles starting from a "seed" profile. In the second task, web crawling techniques can be used to download profile pages and then scrape their content. This allows the adversary to parse the connections lists of user profiles, such as the "friends list" in Facebook, along with their profile information. After that, the adversary can gradually construct the corresponding social graph with all accessible social attributes using a breadth-first search [25]. The adversary can build either a customized web crawler for this task or resort to cheap commercial crawling services that support social-content crawling such as `80legs.com`.

2.3.4 Exploitable Platforms and APIs

Most OSNs provide software APIs that enable the integration of their platforms into third-party software systems. For example, Facebook Graph API [1] enables third parties to read from and write data into Facebook, and provides a simple and consistent view of Facebook's social graph by uniformly representing objects (e.g., profiles, photos) and the connections between them (e.g., friendships, likes, tags). An adversary, however, can use such APIs to automate the execution of online social activities. If an activity is not supported by the API, then the adversary can scrape the content of the platform's web pages, and record the exact HTTP requests which are used to carry out such an activity (i.e., HTTP-request templates). In particular, sending connection requests is often not supported, and is protected against automated usage by CAPTCHAs. This is also the case if a user sends too many requests in a short time period. An adversary, however, can always choose to reduce the frequency at which he sends the requests to avoid CAPTCHAs. Another technique is to inject artificial connection requests into normal OSN traffic at the HTTP level, so that it would appear as if the users added the adversary as a friend [19].

3. THE SOCIALBOT NETWORK

We first start with a conceptual overview of a Socialbot Network (SbN), and briefly outline the adversarial objectives behind maintaining such a network. This is followed by a discussion on the SbN design goals, after which we outline its construction details.

3.1 Overview

We define a *Socialbot Network* (SbN) as a set of socialbots that are owned and maintained by a human controller called the *botherder* (i.e., the adversary). An SbN consists of three components: socialbots, a botmaster, and a Command & Control (C&C) channel. Each socialbot controls a profile in a targeted OSN, and is capable of executing commands that result in operations related to social interactions (e.g., posting a message) or the social structure (e.g., sending a connection request). These commands are either sent by the botmaster or predefined locally on each socialbot. All data collected by the socialbots are called the *botcargo* and are always sent back to the botmaster. A *botmaster* is an OSN-independent software controller that the botherder interacts with in order to define and then send commands through the C&C channel. The C&C *channel* is a communication

Table 1: The generic operations supported by a socialbot in any given OSN.

Operation	Type	Description
`read`(o, p)	Social-interaction	Reads an object o from profile p and returns its value v as botcargo
`write`(v, o, p)	Social-interaction	Writes value v to object o on profile p
`connect`(b, p)	Social-structure	Sends or accepts a connection request sent from profile b to profile p
`disconnect`(b, p)	Social-structure	Breaks the social connection between profiles b and p

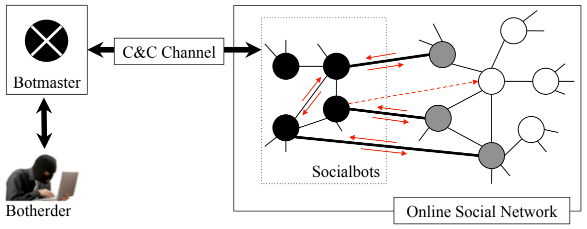

Figure 1: A Socialbot Network. Each node in the OSN represents a profile. The socialbots are marked in black. Infiltrated profiles are marked in gray. Edges between nodes represent social connections. The dashed arrow represent a connection request. The small arrows represent social interactions.

channel that facilitates the transfer of both the botcargo and the commands between the socialbots and the botmaster, including any heartbeat signals. Figure 1 shows a conceptual model of an SbN.

3.2 Objectives

The botherder is a person or an organization that builds and operates an SbN for two main objectives: (1) to carry out a large-scale infiltration campaign in the targeted OSN, and (2) to harvest private users' data. The first objective involves connecting to a large number of either random or targeted OSN users for the purpose of establishing an influential position or fame. The second objective, on the other hand, aims to generate profit by collecting personal users' data that have monetary value. Notice that this data can be then used to craft personalized messages for subsequent spam, phishing, or astroturf campaigns.

3.3 Design Goals

Ideally, an SbN has to be fully automated and scalable enough to control hundreds of socialbots. This is achieved by adopting a traditional web-based botnet design. In order to be effective, however, an SbN has to meet three challenging goals: (1) each socialbot has to be designed in such a way that hides its true face; a robot, (2) the botmaster has to implement heuristics that enable large-scale infiltration in the targeted OSN, and (3) the traffic in the C&C channel has to look benign in order to avoid detecting the botmaster.

In this paper, we decided to use a simplistic design in order to meet each one of these goals. We used techniques that have shown to be both feasible and effective. We discuss the details of these techniques in the following section.

3.4 Construction

We now discuss how a botherder can construct an SbN that performs well in practice while meeting the design goals outlined in the previous section.

3.4.1 *The Socialbots*

A socialbot consists of two main components: a profile on a targeted OSN (the face), and the socialbot software (the brain). We enumerate the socialbots with the profiles they control, that is, for a set $\mathcal{B} = \{b_1, \ldots, b_n\}$ of n socialbots, we use $b_i \in \mathcal{B}$ to refer to both the i-th socialbot and the profile it controls. But how should the socialbot software be programmed in order to mimic real users?

First, we require the socialbot to support two types of generic operations in any given OSN: social-interaction operations that are used to read and write social content, and social-structure operations that are used to alter the social graph. A description of these operations is shown in Table 1.

Second, we define a set of commands that each includes a sequence of generic operations. Each command is used to mimic a real user action that relates to social content generation (e.g., a status update) or social networking (e.g., joining a community of users). Commands can be either defined locally on each socialbots (called *native commands*), or sent by the botmaster through the C&C channel (called *master commands*). For example, we define a native command called `status_update` as follows: at arbitrary times, a socialbot $b_i \in \mathcal{B}$ generates a message m (e.g., a random blurb crawled from the Web), and executes the operation `write`(m, o, b_i) where o is the object that maintains messages on profile b_i (e.g., the profile's "wall" in Facebook).

Finally, each socialbot employs a *native controller*: a simple two-state Finite-State Machine (FSM) that enables the socialbot to either socialize by executing commands, or stay dormant.

3.4.2 *The Botmaster*

A botmaster is a botherder-controlled automation software that orchestrates the overall operation of an SbN. The botmaster consists of three main components: a botworker, a botupdater, and a C&C engine. The *botworker* builds and maintains socialbots. Building a new socialbot involves first creating a new socially attractive profile in the targeted OSN as discussed in Section 2.3.2. After that, the profile's credentials (i.e., the user name and password) are delegated to the socialbot software so as to get a full control over this profile. The *botupdater* pushes new software updates, such as a new list of native commands, to the socialbots through the C&C channel. Finally, the C&C *engine* maintains a repository of master commands and runs a *master controller*: a many-state FSM that is the core control component of the SbN. The botherder interacts with the C&C engine to define a set of master commands, which are dispatched when needed by the master controller and then sent to the socialbots. An interesting question now follows: what kinds of master commands are required to achieve a large-scale infiltration in the targeted OSN?

First, notice that at the beginning each socialbot is isolated from the rest of the OSN, that is, $|\Gamma(b_i)| = 0$ for each

Table 2: **Master commands. The socialbot $b_i \in \mathcal{B}$ is the socialbot executing the command, $|\mathcal{B}| = n$.**

Command	Description
`cluster`	Connects b_i to at most $N_{\texttt{avg}}$ other socialbots in $\mathcal{B}$
`rand_connect(`k`)`	Connects b_i to k non-boherder-owned profiles that are picked at random from the OSN
`decluster`	Disconnects b_i from every socialbot $b_j \in \mathcal{S}$ where $\mathcal{S} = \{b_j \mid b_j \in \Gamma(b_i) \cap \mathcal{B} \text{ and } \lvert\Gamma(b_j)\rvert > n\}$
`crawl_extneighborhood`	Returns $\Delta(b_i)$, the extended neighborhood of b_i, as botcargo
`mutual_connect`	Connects b_i to every profile $p_j \in \Delta(b_i) - \mathcal{B}$.
`harvest_data`	Reads all accessible information of every profile $p_j \in \Gamma(b_i)$, and returns it as botcargo

$b_i \in \mathcal{B}$, which is not a favorable structure to start a large-scale infiltration. Tong et al. [39] show that the social attractiveness of a profile in an OSN is highly correlated to its neighborhood size, where the highest attractiveness is observed when the neighborhood size is close to the network's average. Usually, $N_{\texttt{avg}}$ is known or can be estimated (e.g., $N_{\texttt{avg}} = 130$ on Facebook [3]). Thus, in order to increase the social attractiveness of a socialbot, the adversary defines a master command `cluster`, which orders each socialbot to connect to at most $N_{\texttt{avg}}$ other socialbots.

Second, it has been widely observed that if two users have a mutual connection in common, then there is an increased likelihood that they become connected themselves in the future [22]. This property is known as the *triadic closure principle*, which originates from real-life social networks [32]. Nagle et al. [29] show that the likelihood of accepting a connection request in an OSN is about three times higher given the existence of some number of mutual connections. Therefore, in order to improve the potential infiltration in the targeted OSN, the adversary defines a master command `mutual_connect`, which orders each socialbot to connect to user profiles with whom it has mutual connections.

Finally, we design the master controller to switch between three master states or phases: setup, bootstrapping, and propagation. In the *setup* phase, the botmaster builds n socialbots, updates their software, and then issues the `cluster` command. After that, in the *bootstrapping* phase, the botmaster issues the command `rand_connect(`k`)`, which orders each socialbot to connect to k profiles that are picked at random from the targeted OSN. When every socialbot is connected to k non-botherder-owned profiles, the botmaster issues the command `decluster`, which orders the socialbots to break the social connections between them, and hence, destroying any n-clique structure that could have been created in the earlier step. In the *propagation* phase, the botmaster issues the command `crawl_extneighborhood`, which orders the socialbots to crawl their extended neighborhoods, after which the botmaster uses this information and issues the command `mutual_connect`. Whenever a socialbot infiltrates a user profile, the botmaster issues the command `harvest_data`, which orders the socialbot to collect all accessible users' profile information in its neighborhood. A description of all master commands is shown in Table 2.

3.4.3 The C&C Channel

The communication model of an SbN consists of two channels: the C&C channel and the socialbot-OSN channel. The socialbot-OSN channel carries only OSN-specific API calls and normal HTTP traffic, which are the end product of executing a command by a socialbot. From the OSN side, this traffic originates from either an HTTP proxy in case of high activity, or from a normal user. It is therefore quite difficult to identify a socialbot solely based on the traffic in the socialbot-OSN channel.

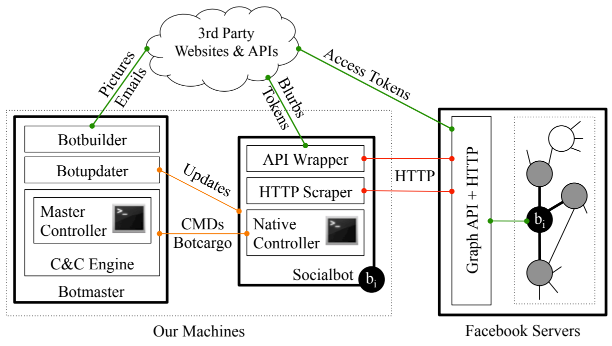

Figure 2: The Facebook Socialbot Network.

As for the C&C channel, how should it be built so that it is particularly hard to identify the botmaster? To start with, we argue that detecting the botmaster from the C&C traffic is as hard as it is in a traditional botnet; the botherder can rely on the existing botnet infrastructure and deploy the SbN as part of the botnet. Alternatively, the botherder can employ advanced techniques that, for example, establish a probabilistically unobservable communication channel by building a covert OSN botnet [28].

4. EVALUATION

In order to evaluate how vulnerable OSNs are to a large-scale infiltration by an SbN, we decided to build one according to the discussion in Section 3.4. We chose Facebook as a targeted OSN because we believe it is particularly difficult to operate an SbN in Facebook for the following reasons: (1) unlike other OSNs, Facebook is mostly used to connect to offline friends and family but not to strangers [13], and (2) Facebook employs the Facebook Immune System (FIS): an adversarial learning system which represents a potential nemesis of any SbN [36].

4.1 Ethics Consideration

Given the nature of an SbN, a legitimate question follows: is it ethically acceptable and justifiable to conduct such a research experiment? We believe that minimal-risk realistic experiments are the only way to reliably estimate the feasibility of an attack in real-world. These experiments allow us, and the wider research community, to get a genuine insight into the ecosystem of online attacks, which are useful in understanding how similar attacks may behave and how to defend against them. This seems to be the opinion of other researchers who share our belief [6, 20].

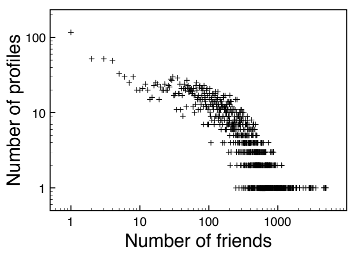

Figure 3: Degree distribution of the generated random sample of Facebook user profiles.

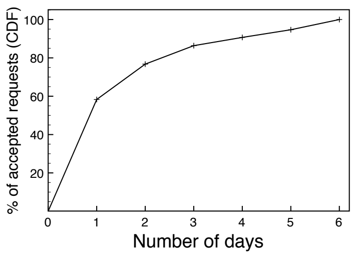

Figure 4: Cumulative distribution of number of days and accepted friendship requests.

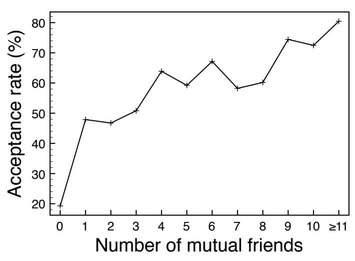

Figure 5: Overall infiltration as a function of number of mutual friends.

We carefully designed our experiment in order to reduce any potential risk at the user side by following known practices [7], and got the approval of our university's behavioral research ethics board. We strongly encrypted and properly anonymized all collected data, which we have completely deleted after we finished our planned data analysis.

4.2 The Facebook SbN

Figure 2 shows the architecture of the SbN we developed. Each socialbot ran the same software and was equipped with only one native command; `status_update`. We implemented the generic operations described in Table 1 using two techniques: API calls and HTTP-request templates, which we now briefly describe. First, we exploited Facebook's Graph API [1] to carry out the social-interaction operations. The API, however, requires the user (i.e., the socialbot in this case) to be logged in to Facebook at the time of any API call. To avoid this, we developed a Facebook application that fetches permanent OAuth 2.0 access tokens that allow each socialbot to send API calls without the need to login. Second, for the social-structure operations, we used pre-recorded HTTP-request templates that allow each socialbot to send friendship requests as if they were sent from a browser. We used an API provided by `iheartquotes.com` to pull random quotes and blurbs which we used as messages for the status updates. As for the botmaster software, we implemented the botworker to interface with three useful websites: `decaptcher.com`; a CAPTCHA-breaking business, `hotornot.com`; a photo-sharing website, and `mail.ru`; an email provider. We also implemented the botupdater with an enhanced functionality to update the HTTP-request templates, along with any new native commands. Finally, we implemented all master commands described in Table 2.

The master command `rand_connect` requires some extra attention. On Facebook, each profile has a unique ID that is represented by a 64-bit integer and is assigned at the time the profile is created. In order to get a uniform sample of Facebook profiles, we decided to use a simple random sampling technique called *rejection sampling* [34], which we now descirbe. First, we generated 64-bit integers at random, but with a range that is reduced to the known ID ranges used by Facebook [15]. Next, we tested whether each generated ID mapped to a real profile by probing the profile page using this ID. Finally, if the profile existed, we included the profile ID in the random sample only if this profile was not isolated. We define an *isolated* user profile as a profile that does not display its "friends list" or has no friends of Facebook.

We deployed the simple two-state native controller and the three-phase, many-state master controller. We acknowledge, however, that more sophisticated controllers could be used that, for instance, employ some machine learning algorithms in order to improve the potential infiltration.

4.3 Operating the Facebook SbN

We operated the Facebook SbN for 8 weeks. The socialbots were able to send a total of 8,570 friendship requests, out of which 3,055 requests were accepted by the infiltrated users. We divide the following discussion according to the three phases of the master controller.

4.3.1 Setup

We created 102 socialbots and a single botmaster, all of which are physically hosted on one machine for simplicity. A botherder, however, could resort to a more sophisticated deployment such as a P2P overlay network. Even though we could have built the socialbots automatically using the botworker, we decided to create them manually as we had no intention to support any CAPTCHA-breaking business. In total, we created 49 socialbots that had male user profiles (referred to as m-socialbots), and 53 socialbots that had female user profiles (referred to as f-socialbots).

4.3.2 Bootstrapping

The socialbots generated a random sample of $5,053$ valid profile IDs. These IDs passed the inclusion criteria we discussed in Section 4.2. Figure 3 shows the degree distribution of this sample.[4]

Based on a pilot study, we decided to send 25 friendship requests per socialbot per day in order to avoid CAPTCHAs. The socialbots took 2 days to send friendship requests to all of the $5,053$ profiles. In total, exactly $2,391$ requests were sent from m-socialbots and $2,662$ from f-socialbots. We kept monitoring the status of the requests for 6 days. Overall, 976 requests were accepted with an average acceptance rate of 19.3%. In particular, 381 of the accepted requests were sent from m-socialbots (15.9% acceptance rate), and 595 were sent from f-socialbots (22.3% acceptance rate). About 86% of the infiltrated profiles accepted the requests within the first three days of the requests being sent, as shown in Figure 4. Overall, the SbN spent two weeks in the bootstrapping phase. For most of that time, however, the SbN was setting idle.

[4]The *degree* of a node is the size of its neighborhood, and the *degree distribution* is the probability distribution of these degrees over the whole network (or a sample of it).

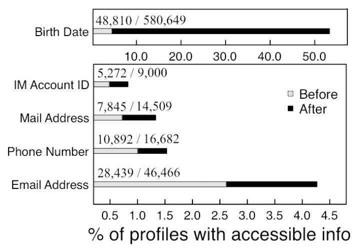

Figure 6: Data revelation of selected profile info before and after a large-scale infiltration.

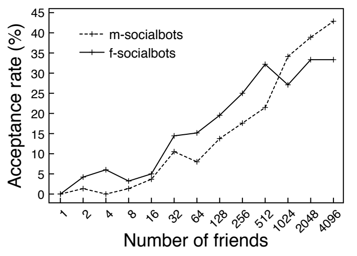

Figure 7: Infiltration as a function of number of friends a user profile has.

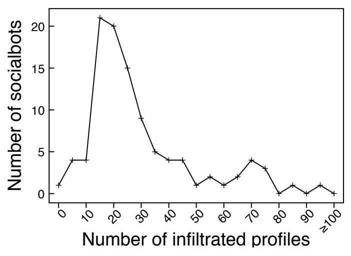

Figure 8: The distribution of number of infiltrated profiles among the socialbots.

4.3.3 Propagation

We kept the SbN running for another 6 weeks. During this time, the socialbots added 3,517 more user profiles from their extended neighborhoods, out of which 2,079 profiles were successfully infiltrated. This resulted in an average acceptance rate of 59.1%, which, interestingly, depends on how many mutual friends the socialbots had with the infiltrated users, and can increase up to 80% as shown in Figure 5.

By the end of the eighth week, we decided to take the SbN down as it resulted in a heavy traffic with Facebook. In total, the SbN generated approximately 250GB inbound and 3GB outbound traffic. We consider the operation time a conservative estimate of the real performance of the SbN as we paused it several times for debugging and data analysis, especially during the bootstrapping phase. We believe that operating the SbN for a longer time is expected to increase the average acceptance rate as the propagation phase will have a higher contribution.

4.4 Harvested Data

As the socialbots infiltrated Facebook, they harvested a large set of users' data. We were able to collect news feeds, users' profile information, and "wall" messages. We decided, however, to only focus on users' data that have monetary value such as Personally Identifiable Information (PII).

After excluding all remaining friendships between the socialbot, the total size of all direct neighborhoods of the socialbots was 3,055 profiles. The total size of all extended neighborhoods, on the other hand, was as large as 1,085,785 profiles. In Table 3, we compare users' data revelation of some PII before and after operating the SbN, as a percentage of the neighborhoods size.

To emphasize its significance, we visualize the data revelation difference of selected profile information in Figure 6. We include all user profiles from *both* the direct and the extended neighborhoods of the socialbots, which added up to 1,088,840 profiles. Each bar in the figure is annotated with two numbers in x/y format, where x and y represent the number of profiles with accessible profile information before and after a large-scale infiltration, respectively.

5. DISCUSSION

In what follows, we discuss the results presented in the previous section and focus on four main points: the observed users' behavior, the effectiveness of the Facebook Immune System, the infiltration performance of the socialbots, and the expected implications on other software systems.

Table 3: Data revelation as % of neighborhoods size.

Neighborhoods	**Direct(%)**		**Extended(%)**	
Profile Info	**Before**	**After**	**Before**	**After**
Gender	69.1	69.2	84.2	84.2
Birth Date	03.5	72.4	04.5	53.8
Married To	02.9	06.4	03.9	04.9
Worked At	02.8	04.0	02.8	03.2
School Name	10.8	19.7	12.0	20.4
Current City	25.4	42.9	27.8	41.6
Home City	26.5	46.2	29.2	45.2
Mail Address	00.9	19.0	00.7	01.3
Email Address	02.4	71.8	02.6	04.1
Phone Number	00.9	21.1	01.0	01.5
IM Account ID	00.6	10.9	00.5	00.8
Average	13.3	34.9	15.4	23.7

5.1 Users' Behavior

Given the results presented in Section 4, someone might ask: are the infiltrated profiles real after all, or are they just other socailbots? To begin with, notice that during the bootstrapping phase, the socialbots targeted profiles that were picked at random out of millions of user profiles, and thus, it is highly unlikely to have picked mostly socialbots.

We also support this argument by the following analysis of the observed users' behavior. First of all, consider Figure 5. The big jump in the acceptance rate from users who were picked at random to those with whom the socialbots had some mutual friends is expected. It directly exhibits the effect of the triadic closure principle, which predicts that having mutual friends will improve the liklihood of accepting a friendship request as discussed in Section 3.4.2. The triadic closure, interestingly, also operated from the users side; the socialbots received a total of 331 friendship requests from their extended neighborhoods.

Second, the behavior depicted in Figure 4 matches the official statistics about real users on Facebook: 50% of the 750 million active Facebook users log on in any given day [3], and thus, it is expected that approximately half of the accepted friendship requests are observed within one day of the requests being sent.

Third and last, the users who were infiltrated during the bootstrapping phase, that is, those who were selected at random, showed another expected behavior [39]: the more friends they had, the higher the chance was that they accepted a friendship request from a socialbot (i.e., a stranger), as shown in Figure 7.

Table 4: SbN *new* data collection performance.

# Profiles Per	Request	Bot	Bot, Day
Gender	0.00	0.05	0.00
Birth Date	62.06	5,214.11	93.11
Married To	1.33	111.58	1.99
Worked At	0.42	35.54	0.63
School Name	10.66	896.05	16.00
Current City	17.61	1,479.87	26.43
Home City	20.29	1,704.99	30.45
Mail Address	0.78	65.33	1.17
Email Address	2.10	176.74	3.16
Phone Number	0.68	56.76	1.01
IM Account ID	0.44	36.55	0.65
Total	116.37	9,777.57	174.60

5.2 SbN vs. Facebook Immune System

The Facebook Immune System (FIS) is a coherent, scalable, and extensible real-time adversarial learning system that is deployed by Facebook to protect its users and the social graph from malicious activities [36]. It performs real-time checks and classification on every read and write action on Facebook database. An interesting question now follows: how did the FIS perform against the SbN we have operated?

After operating the SbN for 8 weeks, only 20 profiles used by the socialbots were blocked by the FIS, and interestingly, all of them were controlled by f-socialbots. After further investigation, we found that these profiles were blocked because some Facebook users flagged them as spam.[5] In fact, we did not observe any evidence that the FIS detected what was really going on other than relying on users' feedback, which seems to be an essential but potentially dangerous component of the FIS.

In reaction, we asked ourselves: what assumptions are made by the FIS that might be problematic? The answer came directly from the authors of the FIS: they state that "fake accounts have limited virality because they are not central nodes in the graph and lack trusted connections. They also have no unique data or history" [36]. Hence, we conjecture that the FIS does not consider fake accounts as a real threat. Fake accounts, however, are one of the main OSN vulnerabilities that allow a botherder to run a large-scale infiltration campaign. Detecting and blocking such accounts—as early as possible—is the main challenge that OSN security defenses like the FIS have to overcome in oder to win the battle against an SbN.

Finally, we noticed that employing the commands `cluster` and `status_update`, which we describe in Table 2, had a desirable effect. It appears that the socialbots seemed more human-like as only 20% of the 102 profiles they controlled got blocked, as apposed to 93% of the 15 profiles we used in our pilot study where we decided not to use these commands. This, in a sense, reflects one of the drawbacks of relying on users' feedback.

5.3 Infiltration Performance

One way to judge whether the resulting infiltration was the making of a small number of "outstanding" socialbots is to compare them in terms of the number of profiles they have infiltrated, as shown in Figure 8. Accordingly, we group the socialbots into two leagues representing the two humps in the figure. The first one constitutes 86% of the socialbots and 70% of the overall infiltration, where each socialbot has infiltrated 0–50 user profiles. The second league, on the other hand, constitutes 10% of the socialbots and 23% of the overall infiltration, where each socailbot has infiltrated 60–80 user profiles. The rest of the socialbots, which we consider as outliers, constitute only 4% of the socialbots with 7% of the overall infiltration.

As most of the resulted infiltration was the outcome of the socialbots' group work, we decided to calculate how much *new* data a botherder can collect per socialbot, as opposed to a completely public access. This is particularly useful when trying to estimate how many socialbots (or connection requests) are needed in order to collect a targeted amount of specific users' data, such as email addresses and birth dates. Using a conservative SbN operation of 25 friendship requests per socialbot per day, and a relaxed operation time of 8 weeks, we found that a botherder is expected to collect an average of 175 new chunks of users' data in Facebook per socialbot per day, as shown in Table 4.

5.4 Implications on other Systems

Operating the SbN for an extended period of time is expected to result in a significantly larger infiltration, as the socialbots will spend most of that time in the propagation phase. Accordingly, the neighborhood size of each socialbot is expected to keep increasing. This, however, has alarming implications on software systems that use the social graph of an OSN to provide services such as limiting Sybil nodes in distributed systems [45] and modeling trust in socially-informed recommender systems [43].

To explain why this is particularly important, let us consider OSN-based Sybil defenses used in P2P overlay networks.[6] In general, such a defense mechanism uses two types of networks that share the same nodes: the P2P network which we try to protect, and an external social network representing a trust network such as Facebook [45]. Now, in order to detect Sybil nodes in the P2P network, the following assumption is often made [41]: a Sybil node cannot have many connections with non-Sybil nodes in the social network. Thus, finding a well-connected node in the P2P network, which is loosely connected in the social network, is a good indication that this node is a Sybil. This assumption, however, is not safe. We showed that the socialbots can have arbitrarily many social connections with arbitrary OSN users. Therefore, we believe that using a social network such as Facebook to detect Sybil nodes in P2P networks is, in fact, ineffective. This conclusion extends to all systems that make this assumption about OSNs.

6. RELATED WORK

The closest threat to large-scale infiltration in OSNs is a botnet called Koobface [5]. At first, Koobface compromises user accounts on OSNs, and then uses these accounts to promote a provocative message with a hyperlink. The link points to a phishing website that asks the user to install

[5]Based on the content of a pop-up message that Facebook showed when we manually logged in.

[6]A Sybil node (or a Sybil for short) is an adversary-owned online identity, account, profile, or endpoint in a particular network. Sybil nodes represent a serious threat and can be used to subvert services such routing in P2P networks [11].

a Flash plugin which is, in fact, the Koobface executable. Unlike Koobface, an SbN does not rely on hijacked profiles as doing so requires infecting a large number of initial "zombie" machines through OSN-independent distribution channels.

Bilge et al. [6] show that most users in OSNs are not cautious when accepting connection requests that are sent to them. The authors did an experiment to test how willing users are to accept connection requests from forged user profiles of people who were already in their friendship list as confirmed contacts. They also compared that with users' response to connection requests from people that they do not know (i.e., fake profiles representing strangers). In their experiment, they show that the acceptance rate for forged profiles was always over 60%, and about 20% for the fake profiles. Unlike their targeted attack, we do not expect the adversary to forge profiles as this limits the scalability of the attack and makes it more susceptible to detection.

Other than the wide botnet literature, the majority of the work we found relates to Twitter bots and different techniques to detect them. Ratkiewicz et al. [33] describe the use of Twitter bots to spread misinformation in the run-up to the U.S. political elections. Stringhini et al. [37] analyze to what extent spam has entered OSNs, and how spammers who target these OSNs operate. The authors develop techniques to detect spammers, and show that it is possible to automatically identify the accounts they use. Grier et al. [16] show that 8% of the unique URLs that are sent in Twitter are later blacklisted. The authors also described a test on Tweets' timestamps to identify automated Twitter accounts, or spambots, by regularities in the times when they tweet, which they have found to have a high accuracy in identifying Twitter spammers.

7. CONCLUSION

We have evaluated how vulnerable OSNs are to a large-scale infiltration by a Socialbot Network (SbN). We used Facebook as a representative OSN, and found that using bots that mimic real OSN users is effective in infiltrating Facebook on a large scale, especially when the users and the bots share mutual connections. Moreover, such socialbots make it difficult for OSN security defenses, such as the Facebook Immune System, to detect or stop an SbN as it operates. Unfortunately, this has resulted in alarming privacy breaches and serious implications on other socially-informed software systems. We believe that large-scale infiltration in OSNs is only one of many future cyber threats, and defending against such threats is the first step towards maintaining a safer social Web for millions of active web users.

8. ACKNOWLEDGMENTS

We would like to thank San-Tsai Sun, Elizeu Santos-Neto, Albina Muslukhova, and Bader AlAhmad for their kind help and advice. We also would like to thank Cormac Herley, Miranda Mowbray, and Adriana Iamnitchi for their feedback on an early draft of this paper. This research is partially supported through funding from the NSERC Internetworked Systems Security Network (ISSNet).

9. REFERENCES

[1] Facebook Open Graph Protocol. `http://developers.facebook.com/docs/opengraph`.

[2] Sick profile maker. `http://sickmarketing.com/sick-profile-maker`.

[3] Facebook Statistics. `http://www.facebook.com/press`, March 2011.

[4] Jet bots. `http://allbots.info`, 2011.

[5] J. Baltazar, J. Costoya, and R. Flores. The real face of Koobface: The largest web 2.0 botnet explained. *Trend Micro Research*, July 2009.

[6] 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: Proceedings of the 18th International Conference on World Wide Web*, pages 551–560, New York, NY, USA, 2009. ACM.

[7] N. Bos, K. Karahalios, M. Musgrove-Chávez, E. S. Poole, J. C. Thomas, and S. Yardi. Research ethics in the facebook era: privacy, anonymity, and oversight. In *CHI EA '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 2767–2770, New York, NY, USA, 2009. ACM.

[8] D. Boyd. Social media is here to stay... Now what? *Microsoft Research Tech Fest*, February 2009.

[9] G. Brown, T. Howe, M. Ihbe, A. Prakash, and K. Borders. Social networks and context-aware spam. In *CSCW '08: Proceedings of the ACM 2008 Conference on Computer Supported Cooperative Work*, pages 403–412, New York, NY, USA, 2008. ACM.

[10] Z. Coburn and G. Marra. Realboy: Believable twitter bots. `http://ca.olin.edu/2008/realboy`, April 2011.

[11] J. R. Douceur. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.

[12] M. Egele, L. Bilge, E. Kirda, 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*, pages 1865–1870, New York, NY, USA, 2010. ACM.

[13] N. B. Ellison, C. Steinfield, and C. Lampe. The benefits of Facebook "friends:" social capital and college students' use of online social network sites. *Journal of Computer-Mediated Communication*, 12(4):1143–1168, July 2007.

[14] N. FitzGerald. New Facebook worm - don't click da' button baby! `http://fitzgerald.blog.avg.com/2009/11/`, November 2009.

[15] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in Facebook: A case study of unbiased sampling of osns. In *Proceedings of IEEE INFOCOM '10*, San Diego, CA, March 2010.

[16] 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*, CCS '10, pages 27–37, New York, NY, USA, 2010. ACM.

[17] C. Herley. The plight of the targeted attacker in a world of scale. In *The 9th Workshop on the Economics of Information Security (WEIS 2010)*, 2010.

[18] C. Hernandez-Castro and A. Ribagorda. Remotely

telling humans and computers apart: An unsolved problem. In J. Camenisch and D. Kesdogan, editors, *iNetSec 2009 ? Open Research Problems in Network Security*, volume 309 of *IFIP Advances in Information and Communication Technology*, pages 9–26. Springer Boston, 2009.

[19] M. Huber, M. Mulazzani, and E. Weippl. Who on earth is Mr. Cypher? automated friend injection attacks on social networking sites. In *Proceedings of the IFIP International Information Security Conference 2010: Security & Privacy — Silver Linings in the Cloud*, 2010.

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

[21] A. M. Kaplan and M. Haenlein. Users of the world, unite! the challenges and opportunities of social media. *Business Horizons*, 53(1):59 – 68, 2010.

[22] J. Leskovec and E. Horvitz. Planetary-scale views on a large instant-messaging network. In *Proceeding of the 17th International Conference on World Wide Web*, pages 915–924, New York, NY, USA, 2008. ACM.

[23] G. Livingston. Social media: The new battleground for politics. `http://mashable.com/2010/09/23/congress-battle-social-media/`, September 2010.

[24] D. Misener. Rise of the socialbots: They could be influencing you online. `http://www.cbc.ca/news/technology/story/2011/03/29/f-vp-misener-socialbot-armies-election.html`, March 2011.

[25] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *IMC '07: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, pages 29–42, New York, NY, USA, 2007. ACM.

[26] E. Morozov. Swine flu: Twitter's power to misinform. `http://neteffect.foreignpolicy.com/posts/2009/04/25/swine_flu_twitters_power_to_misinform`, April 2009.

[27] 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*, USENIX Security'10, pages 28–28, Berkeley, CA, USA, 2010. USENIX Association.

[28] S. Nagaraja, A. Houmansadr, P. Agarwal, V. Kumar, P. Piyawongwisal, and N. Borisov. Stegobot: A covert social network botnet. In *Proceedings of the Information Hiding Conference*, 2011.

[29] F. Nagle and L. Singh. Can friends be trusted? exploring privacy in online social networks. In *Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining*, pages 312–315, Washington, DC, USA, 2009. IEEE Computer Society.

[30] S. Patil. Social network attacks surge. `http://www.symantec.com/connect/blogs/social-network-attacks-surge`, June 2011.

[31] A. Pitsillidis, K. Levchenko, C. Kreibich, C. Kanich, G. M. Voelker, V. Paxson, N. Weaver, and S. Savage. Botnet judo: Fighting spam with itself. In *NDSS*, 2010.

[32] A. Rapoport. Spread of information through a population with socio-structural bias: I. assumption of transitivity. *Bulletin of Mathematical Biology*, 15:523–533, 1953. 10.1007/BF02476440.

[33] 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 *Proceedings of the 20th international conference companion on World wide web*, WWW '11, pages 249–252, New York, NY, USA, 2011. ACM.

[34] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[35] F. Salem and R. Mourtada. Civil movements: The impact of Facebook and Twitter. *The Arab Social Media Report*, 1(2), 2011.

[36] T. Stein, E. Chen, and K. Mangla. Facebook immune system. In *Proceedings of the 4th Workshop on Social Network Systems*, SNS '11, pages 8:1–8:8, New York, NY, USA, 2011. ACM.

[37] G. Stringhini, C. Kruegel, and G. Vigna. Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 1–9, New York, NY, USA, 2010. ACM.

[38] C. Taylor. Why not call it a Facebook revolution? `http://edition.cnn.com/2011/TECH/social.media/02/24/facebook.revolution/`, February 2011.

[39] S. T. Tong, B. Van Der Heide, L. Langwell, and J. B. Walther. Too much of a good thing? the relationship between number of friends and interpersonal impressions on Facebook. *Journal of Computer-Mediated Communication*, 13(3):531–549, 2008.

[40] J. A. Vargas. Obama raised half a billion online. `http://voices.washingtonpost.com/44/2008/11/obama-raised-half-a-billion-on.html`, November 2008.

[41] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove. An analysis of social network-based sybil defenses. In *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, SIGCOMM '10, pages 363–374, New York, NY, USA, 2010. ACM.

[42] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard AI problems for security. In E. Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311. Springer, 2003.

[43] F. Walter, S. Battiston, and F. Schweitzer. A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems*, 16:57–74, 2008.

[44] H. Yeend. Breaking CAPTCHA without OCR. `http://www.puremango.co.uk/2005/11/breaking_captcha_115/`, November 2005.

[45] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 3–17, Washington, DC, USA, 2008. IEEE Computer Society.