The Socialbot Network: When Bots Socialize for Fame and Money

Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, Matei Ripeanu

Laboratory for Education and Research in Secure Systems Engineering
Networked Systems Laboratory

University of British Columbia, Vancouver, Canada

Technical report LERSSE-REPORT-2012-001*

Last Modification Date: 2012/02/27

Revision: #3

Abstract

Online Social Networks (OSNs) have attracted millions of active users and have become an integral part of today's Web ecosystem. Unfortunately, in the wrong hands, OSNs can be used to harvest private user data, distribute malware, control botnets, perform surveillance, influence algorithmic trading, and spread misinformation. Usually, an adversary starts off by running an infiltration campaign using hijacked or adversary-owned OSN accounts, with an objective to connect to a large number of users in the targeted OSN. In this paper, we evaluate how vulnerable OSNs are to a large-scale infiltration by *socialbots*: bots that control OSN accounts and mimic actions of real users. We adopted a traditional web-based botnet design and built a prototype of a *Socialbot Network* (SbN): a group of coordinated programmable socialbots. We operated our prototype on Facebook for eight weeks, and collected data about users' behavior in response to a large-scale infiltration by our socialbots. 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, 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.

Contents

1 Introduction

Online Social Networks (OSNs) such as Facebook[1] and Twitter[2] have far exceeded their original goal 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 [36, 41]. 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 the digital era in presidential fundraising [69]. Similarly, 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 [58, 62]. This pervasive integration of social media into everyday life is rapidly becoming the norm, and arguably is here to stay [11]. Today's online social experience, however, is not exclusive to only human beings.

A new breed of computer programs called *socialbots* are now online, and they can be used to influence OSN users [44]. 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) or spambots (e.g., Facebook bots that distribute unsolicited messages to non-consenting users) is that it is designed to pass itself off as a human being. This allows the socialbot to *infiltrate* a targeted OSN in order to reach an influential position, that is, to compromise the social graph by connecting to a large number of its users. This position can be then exploited to influence OSN users [44], spread misinformation and propaganda in order to bias the public opinion [46], perform surveillance, or even influence algorithmic trading in stock markets [6, 9]. For example, Ratkiewicz et al. [56] describe the use of Twitter bots to run astroturf and smear campaigns during the 2010 U.S. midterm elections. Moreover, a socialbot can exploit its new position in the network to promote and distribute malicious content such as botnet executables [73]. For instance, the Koobface botnet [5] propagates by hijacking OSN accounts of infected machines, after which it uses these accounts to send messages with a malicious link to other OSN users. This link points to a legitimate but compromised website

[1] `https://facebook.com`
[2] `https://twitter.com`

that attempts to infect its visitors with the Koobface malware.

As socialbots infiltrate OSN users, they can also harvest private user data, such as email addresses, phone numbers, and other personal information 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 [53]. It is thus not surprising that similar socialbots are being offered for sale in the Internet underground markets, with prices starting from $29 per bot [3].

Recently, a number of techniques have been proposed that aim to automatically identify spambots in OSNs based on their abnormal behavior [26, 54, 61, 74]. For example, Stein et al. [59] 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 how OSN users might behave in response to a large-scale infiltration by such deceptive bots.

In this paper, we aim to fill this knowledge gap. We studied large-scale infiltration in OSNs as an organized campaign run by an army of socialbots to connect to either random or targeted OSN users on a large scale. Therefore, we adopted a traditional web-based botnet design and defined what we call a *Socialbot Network* (SbN): a group of programmable socialbots that are coordinated by an adversary (referred to as a *botherder*) using a software controller (referred to as a *botmaster*). The botmaster was designed to exploit the known properties of social networks, such as the *triadic closure principle* [55], in order to improve the magnitude of the potential infiltration.

We created a fairly small and simple, yet effective, SbN consisting of 102 socialbots and a single botmaster, and then operated this SbN on Facebook for eight weeks. During that time, the socialbots sent a total of 8,570 connection requests, out of which 3,055 were accepted. We recorded all data related to the resulted infiltration by this SbN and the corresponding users' behavior, along with all accessible users' profile information. We summarize our findings in what follows:

(1) *OSNs such as Facebook are vulnerable to large-scale infiltration campaigns.* From the OSN side, we show that today's OSNs exhibit inherent vulnerabilities that allow an adversary to automate the infiltration on a large scale (Sections 3 and 4). From the user side, we show that most OSN users are not careful enough when accepting connection requests, espe-

cially when they share mutual connections with the sender. This behavior can be exploited to achieve a large-scale infiltration with a success rate of up to 80% (Sections 5 and 6).

(2) *Depending on users' privacy settings, operating an SbN can result in serious privacy breaches.* We show that after a large-scale infiltration, a botherder can harvest large amounts of publicly inaccessible user data. This data include email addresses, phone numbers, and other profile information of the infiltrated users, all of which have monetary value. Unfortunately, this also includes the private data of "friends of friends", that is, users who have not been infiltrated but are connected to infiltrated users (Section 6).

(3) *In practice, OSN security defenses such as the FIS are not effective enough in detecting a large-scale infiltration as it occurs.* Our results show that a successful infiltration of an OSN user is expected to be observed within the first three 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 the result of the 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 (Section 6).

In conclusion, our findings shed light on the importance of considering the human factor when designing OSN security defenses. We believe that socio-technical solutions are required to effectively protect the social Web and realize security defenses that are less vulnerable to both human and technical exploits (i.e., automated social engineering and platform hacks, respectively).

2 Background and Preliminaries

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

2.1 Online Social Networks

An *Online Social Networks* (OSN) is a centralized web *platform* that facilitates and mediates users' social activities online. 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 such as friendships in Facebook, or directed such as follower-followee relationships in Twitter.

An OSN can be modeled as a *social 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 $\overline{N}(G) = \eta$, which is defined as follows:

$$\overline{N}(G) = \frac{1}{|V|} \sum_{u \in V} |\Gamma(u)| = \eta.$$

Finally, we call the set $\Delta(u)$ the *extended neighborhood* of u, which is defined as the union of the neighborhoods of all users in $\Gamma(u)$ as follows:

$$\Delta(u) = \bigcup_{v \in \Gamma(u)} \Gamma(v).$$

2.2 Large-Scale Infiltration and Sybil Attacks

The *Sybil attack* refers to the situation where an adversary controls multiple identities and joins a targeted system under these identities many times in order to subvert a particular service [16]. In the Sybil attack, each adversary-controlled identity is called a *Sybil* and is represented by an account or an endpoint, depending on the targeted system. Accordingly, we define *large-scale infiltration* in OSNs as an instance of the Sybil attack where an adversary employs an automation software, which is scalable enough to control many adversary-owned OSN accounts (i.e., Sybils), in order to connect to a large number of users in the in the targeted OSN. We study large-scale infiltration as an organized, adversarial campaign that has two objectives: compromising the social graph of the targeted OSN and collecting users' private data en masse. We discuss these objectives in detail later on in Section 4.2.

Recent research indicates that large-scale infiltration in OSNs is possible [50]. For example, in the context of identity theft, Bilge et al. [8] 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 sent by people 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 infiltration and makes it more susceptible to detection. Moreover, we aim to characterize more descriptive user behaviors that are important to improve today's OSN security defenses, and to evaluate the corresponding security and privacy implications, all under the context of large-scale infiltration. To the best of our knowledge, we present the first comprehensive treatment of this topic.

2.3 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 [4]. 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 [5, 13]. In fact, some of these attacks are sophisticated and use heuristics and learned observations about users' behavior in the targeted system in order to increase the magnitude of their potential damage [5, 8, 34].

Huber et al. [32] present one of the first frameworks for automated social engineering in OSNs, and show that a new breed of bots, which we generally call *socialbots*, can be developed in order to automate traditional social engineering attacks in OSNs for many adversarial objectives. In fact, this automation has a strong economic rationale behind it. Herley [29] shows that for an online attack to be scalable, it ought to be automated without manual per-user adjustments. Otherwise, there are no economic incentives for a rational adversary to scale the attack, which is undesirable from an adversarial standpoint.

In concept, a socialbot is an automation software that controls a profile in a particular OSN, and has the ability to execute basic social activities. What is

special about a socialbot is that it is designed to be stealthy, that is, it is able to pass itself off as a human being. This is achieved by either simply mimicking the actions of a real OSN user or by simulating such a user using artificial intelligence, just as in social robotics [24]. For example, Realboy [14] is one of the first experimental projects that aim to design believable Twitter bots which imitate real Twitter users.

The socialbots can be used for non-adversarial objectives as well. For example, the Web Ecology Project [63] envisions the design of socialbots that have positive impact on online communities by advocating awareness and cooperation among OSN users on civic or humanitarian issues. Furthermore, the Pacific Social Architecting Corporation (PacSocial) employs socialbots for *social architecture* [51]: the technology where advanced socialbots are used to interact with, promote, or provoke online communities towards desirable behaviors, including large-scale restructuring of social graphs.

3 OSN Vulnerabilities

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

3.1 Ineffective CAPTCHAs

OSNs employ CAPTCHAs [70] to prevent automated bots from abusing their platforms. An adversary, however, can often circumvent this countermeasure by using different techniques such as automated analysis via optical character recognition and machine learning [8], exploiting botnets to trick the infected victims into manually solving CAPTCHAs [5, 18], reusing session IDs of known CAPTCHAs [30], cracking MD5 hashes of CAPTCHAs that are validated on the client side [75], or hiring cheap human labor [47].

Let us consider the use of cheap human labor to solve CAPTCHAs; a phenomenon that is known as CAPTCHA-solving business. Motoyama et al. [47] 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 solved 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 study online attacks from an economic context, and define cost metrics and structures that measure when it is economically feasible for an adversary to mount a large-scale attack that involves, for instance, solving CAPTCHAs by employing cheap human labor [29].

3.2 Sybil Accounts and Fake 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. The latter case represents a potential Sybil attack, which we further study in Section 6.4. In what follows, we show that an adversary can fully automate the account creation process in order to create a set of Sybil user accounts, where each account is represented by a fake user profile. This, however, is not new as similar tools are used for online marketing [2, 22]. The adversary can write a customized software to create such accounts or buy OSN accounts in bulk from online forums or freelance websites [12, 48].

3.2.1 Sybil User Accounts

When creating a new user account on 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 Sybil account: providing an active email address that he owns and account activation. To tackle the first hurdle, the adversary can maintain many email addresses by either using "temp" email addresses that are obtained from providers that do not require registration such as 10MinuteEmail[3], or by creating email addresses using email providers that do not limit the number of created email accounts per browsing session or IP address such as MailRu[4]. As

[3] `http://www.10minutemail.com`
[4] `http://www.mail.ru`

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, which is typically included in the downloaded email.

3.2.2 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 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 [8, 23]. 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[5], where users publicly post their personal pictures for others to rate their "hotness".In fact, such websites also provide categorization of the rated personal pictures based on gender and age range. It is thus possible for an adversary to automate the collection of the required profile information in order to populate a fake user profile by crawling, or scavenging in this case, the Web.

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 one or many accounts, 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 an online search algorithm such as breadth-first search [45]. The ad-

[5] `http://www.hotornot.com`

versary 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[6].

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 social activities online. If an activity is not supported by the API, then the adversary can scrape the content of the platform's website, 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 usually 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 [33].

4 The Socialbot Network

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

4.1 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

[6] `http://www.80legs.com`

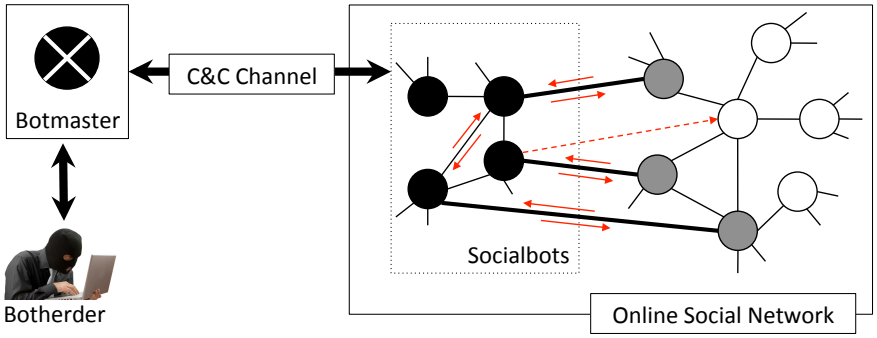


Figure 1: A Socialbot Network (SbN). Each node in the OSN represents a profile. The socialbots are marked in black. Infiltrated profiles are marked in gray. Edges between nodes represent social connections. The dashed arrow represents a connection request. The small arrows represent social interactions. The SbN can be part of an existing botnet, where each "zombie" machine is additionally infected by the socialbot malware that controls a fake user profile in the targeted OSN.

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

4.2 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 a 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, which can then be exploited to promote malicious content or spread misinformation. The second objective, on the other hand, aims to generate profit by collecting private user

data that have monetary value. Notice that these data can be then used to craft personalized messages for subsequent spam, phishing, or astroturf campaigns.

4.3 Threat Model

We assume the threat model of a global passive adversary. Since an SbN can be deployed as part of an existing botnet, we treat it as a distributed network of compromised "zombie" machines acting cooperatively. Accordingly, we believe it is fair to assume that the defenders (i.e., OSNs and ISPs) are also able to cooperate, and hence, have a global view of the communication traffic. We also assume that botnet infections are not easily detected, that is, an SbN cannot tolerate 100% clean up of all infected machines, just like any other botnet. We expect, however, an SbN to tolerate random losses of a large number of compromised machines because at least one machine is required to host all of the socialbots, as we show in Section 5.

4.4 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 article, 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 acknowledge, however, that more sophisticated techniques that utilize machine learning algorithms are possible, but we refrain from using them as our objective is to evaluate the threat of large-scale infiltration and characterize users' behavior, rather than to optimize the performance of an SbN. We discuss the details of the used techniques in the following section.

4.5 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.

4.5.1 The Socialbots

A socialbot consists of two main components: a profile on a targeted OSN (the face), and the socialbot software (the brain). Given that we develop the socialbot software in an adverserial setting, we regard this software as being malicious, and refer to it as malware. We enumerate the socialbots by the profiles they control, that is, for a set $\mathcal{B} = \{b_1, \ldots, b_m\}$ of m 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 malware be programmed in order to mimic real users, at least naïvely?

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 to the socialbots 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). This command resembles an OSN user posting a status update message on her profile, and is executed at arbitrary times in order to avoid creating detectable patterns. Likewise, more sophisticated commands can be defined that, for instance, allow the socialbots to comment on each others' status updates. Moreover, each socialbot can be enhanced with advanced social-interaction capabilities by integrating existing chatterbots, such as Eliza [72], into the socialbot's malware.

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.

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

Table 2: Master commands. The socialbot $b_i \in \mathcal{B}$ is the socialbot executing the command, where $|\mathcal{B}| = m$ and $\overline{N}(G) = \eta$.

Command	Description
`cluster`	Connects b_i to at most η 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 $\mathcal{S} = \{b_j \mid b_j \in \Gamma(b_i) \cap \mathcal{B}$ and $\lvert\Gamma(b_j)\rvert > m\}$
`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

4.5.2 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 3.2. After that, the profile's credentials (i.e., the user name and password) are delegated to the socialbot's malware in order to get a full control over this user profile. If the SbN is operated as part of a botnet, the socialbot malware can use hijacked OSN accounts instead. This, however, might make the socialbot more susceptible to detection [5]. The *botupdater* pushes new software updates, such as new native commands or updated HTTP-request templates, 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 two-fold question now follows: what kinds of master commands are required to achieve a large-scale infiltration in the targeted OSN, and when should they be dispatched by the master controller?

First, notice that at the beginning each socialbot is isolated from the rest of the OSN, that is, $|\Gamma(b_i)| = 0$ for each $b_i \in \mathcal{B}$, which is not a favorable structure to start a large-scale infiltration. Tong et al. [64] 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 $\overline{N}(G) = \eta$. Usually, η is known or can be estimated (e.g., $\eta = 130$ on Facebook [20]). Thus, in order to increase the social attractiveness of a socialbot, the botherder defines a master command `cluster`, which orders each socialbot to connect to at most η other socialbots. Moreover, this might be helpful in order to elude OSN security defenses that keep track of how many rejected connection requests each new user ends up with when joining the OSN for the first time, which is usually used by such systems as an indication of an automated activity or spam [74].

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 [40]. This property is known as the *triadic closure principle*, and it originates from real-life social networks [55]. Nagle et

al. [50] 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 botherder defines a master command `mutual_connect`, which orders each socialbot to connect to user profiles with whom it has mutual connections (i.e., users in the extended neighborhood $\Delta(b_i)$ for each socialbot $b_i \in \mathcal{B}$).

Finally, we design the master controller to switch between three super states or phases: setup, bootstrapping, and propagation. In the *setup* phase, the botmaster builds m socialbots, updates their malware, 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 m-clique structure that could have been created in the earlier step.[7] 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 the crawled 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.

4.5.3 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 it generates in the socialbot-OSN channel.

As for the C&C channel, how should it be built so that it is particularly hard

[7] In more advanced implementation, a socialbot would break one social connection with another socialbot for every newly infiltrated user profile, and thus, *gradually* `decluster`.

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 an 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 [49].

5 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 4.5. We chose Facebook as a target OSN because it is the largest OSN found today, consisting of more than 750 million users [20]. Besides, we believe it is particularly difficult to operate an SbN on Facebook as (1) unlike other OSNs, Facebook is mostly used to connect to real-life friends and family but not to strangers [19, 38, 39], and (2) Facebook employs the Facebook Immune System (FIS) [59]: an adversarial learning system which represents a potential nemesis of any SbN.

5.1 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 controlled, 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 as well [8, 18, 33, 34].

We carefully designed our experiment in order to reduce any potential risk at the user side [10]. In particular, we followed the known practices 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.

As part of our code of ethics, we communicated the details of our experiment to Facebook before any publication, and accordingly, we decided not to include specific technicalities about a set of vulnerabilities we discovered

in Facebook's platform. We believe that these platform vulnerabilities can be exploited by cyber criminals to mount different kinds of online attacks. We, however, reported these vulnerabilities to Facebook through its platform's online vulnerability reporting tool [21].

5.2 Methodology

Our main research objective is to characterize users' response to a large-scale infiltration campaign in OSNs, along with the corresponding security and privacy implications. To that end, we built an SbN prototype targeting Facebook for the reasons outlined above, and operated this SbN for eight weeks during the first quarter of 2011. The duration of the experiment was informed by how many data points we needed to properly capture users' behavior, and accordingly, we took the SbN down once we stopped observing new trends. We report only the results we observed during the length of the experiment. We used a single machine and two types of IP addresses at different stages of the experiment. The first IP address was assigned by the university, and the second IP address was assigned by a commercial ISP. We also implemented a simple HTTP proxy on the machine we used in order to make the traffic looks like as if it originated from multiple clients having different browsers and operating systems. Even though the university assigned IP address *might* have diluted the Facebook Immune System [7], we believe that it is unsafe to completely white-list university IP addresses.[8] In fact, today's botnet owners struggle over who has the largest number of "high-quality" infected machines, including university, corporate, and even government machines [17].

5.3 The Facebook SbN

Figure 2 shows the architecture of the SbN we developed. Each socialbot ran the same malware 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

[8]During the SbN operation using a university IP address, we observed that some actions were identified as malicious, and the used IP address was temporarily blocked, especially during Sybil accounts creation. This supports the argument that even university IP addresses were audited by Facebook, and they were not fully white-listed.

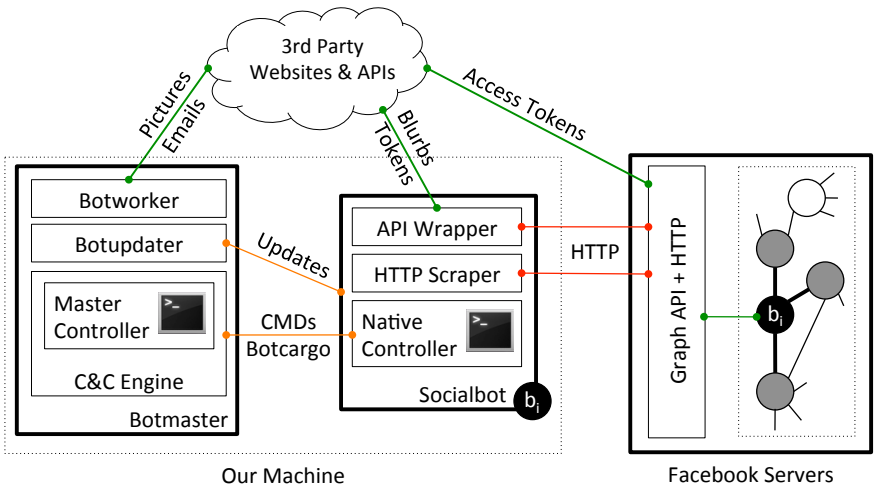


Figure 2: The Facebook Socialbot Network.

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 in order to 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[9] to pull random quotes and blurbs, and used them as messages for the status updates. As for the botmaster software, we implemented the botworker to interface with three useful websites: DeCaptcher[10]; a CAPTCHA-solving business, HotOrNot; a photo-sharing website, and MailRu; an email provider. We also implemented the botupdater with an enhanced functionality to update the HTTP-request template, along with any new native commands. Finally, we implemented all master commands described in Table 2.

The master command `rand_connect(`k`)` requires some extra attention. On Facebook, each profile has a unique identifier that is represented by a 64-bit integer and is assigned at the time the profile is created. In order to get a uniform random sample of Facebook profiles, we decided to use a simple random sampling technique called *rejection sampling* [57], which we now describe. First, we generated 64-bit integers at random, but with a range that

[9] `http://www.iheartquotes.com`
[10] `http://www.decaptcher.com`

is reduced to the known identifier ranges used by Facebook [25]. Next, we tested whether each generated identifier mapped to an existing user profile by probing the profile's page using this identifier. Finally, if the profile existed, we included this profile identifier 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. A more resourceful attacker, however, would employ adversarial classifier reverse engineering techniques [42] in order to learn sufficient information about the security defenses deployed by the targeted OSN, and then construct an adversarial attack that maximizes the potential infiltration and minimizes the detection rate.

5.4 Operating the Facebook SbN

We operated the Facebook SbN for eight weeks during the first quarter of 2011. 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.

5.4.1 Setup

As $\eta = 130$ on Facebook, we decided to create 102 socialbots and a single botmaster, all of which were physically hosted on a single machine for simplicity. A botherder, however, would resort to a more sophisticated deployments such as peer-to-peer overlay networks [27]. 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-solving business. In total, we created 49 socialbots that had male user profiles, to which we refer as m-socialbots, and 53 socialbots that had female user profiles, to which we refer as f-socialbots. As expected, the socialbots clustered into a 102-clique structure, representing a tightly-knit, cohesive community of OSN users that is useful for bootstrapping the infiltration, as discussed in Section 4.5.2

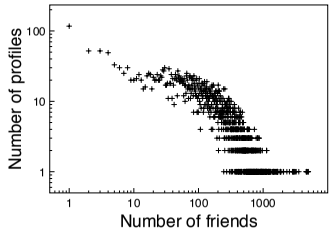


Figure 3: Degree distribution of the generated random sample of Facebook user profiles during the bootstrapping phase, with a sample size of 5,053 valid profile identities.

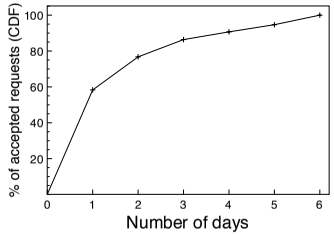


Figure 4: Cumulative distribution function of number of days it took to observe a fraction of the overall accepted friendship requests during the bootstrapping phase.

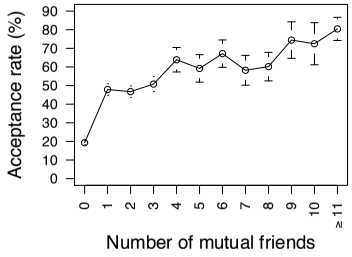


Figure 5: Average acceptance rate of the resulted infiltration as a function of the number of mutual friends the socialbots had with the infiltrated users. (95% conf.)

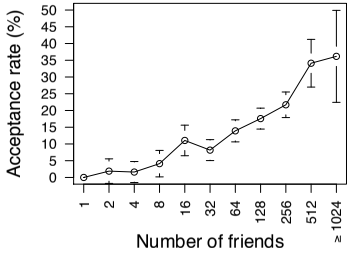


Figure 6: Average acceptance rate as a function of the number of friends a user profile had during the bootstrapping phase. (fall requests sent by m-socialbots, 95% conf.)

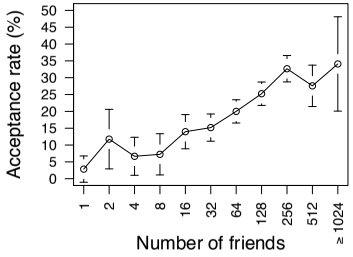


Figure 7: Average acceptance rate as a function of the number of friends a user profile had during the bootstrapping phase. (all requests sent by f-socialbots, 95% conf.)

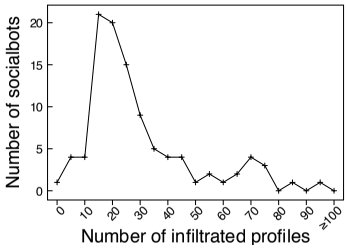


Figure 8: Distribution of the overall infiltration among the socialbots. A point in the figure represents how many socialbots infiltrated the corresponding number of user profiles.

5.4.2 Bootstrapping

The socialbots generated a random sample of 5,053 valid profile identities. These identities passed the inclusion criteria outlined in Section 5.3. Figure 3 shows the degree distribution of this sample.[11]

Based on a pilot study, we decided to send 25 friendship requests per socialbot per day in order to avoid CAPTCHAs. The socialbots took two 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 six 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). The difference in the average acceptance rate was statistically significant ($\chi^2 = 32.8702$, $p = 9.852 \times 10^{-9}$), where the f-socialbots outperformed the m-socialbots by 6.4% on average.[12] About 86% of the infiltrated profiles accepted the requests within the first three days of the requests being sent, as shown in Figure 4. In our implementation, the socialbots gradually broke the 102-clique structure as they infiltrated more and more user profiles. Overall, the SbN spent two weeks in the bootstrapping phase. For most of that time, however, the SbN was setting idle.

5.4.3 Propagation

We kept the SbN running for another six weeks. During that 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 overall 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 we stopped observing new trends in users' behavior. Moreover, the SbN resulted in a heavy traffic with Facebook where we recorded approximately 250GB inbound and 3GB outbound of network traffic. We consider the operation time a conservative estimate of the real performance of the SbN as we paused it sev-

[11] 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).

[12] Using a two-sample test for equality of proportions.

eral times for debugging and data analysis, especially during the bootstrapping phase. For example, Facebook changed the HTTP-request parameters used for sending a friendship request through their platform, and thus, we had to pause the SbN operation in order to record the new parameters, use the botupdater to push the new HTTP-request template to the socialbots, and then continue the SbN operation. 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, as suggested by Figure 5.

6 Discussion

In what follows, we discuss the results presented in the previous section and focus on three main points: the observed users' behavior, the harvested users' data, the infiltration performance of the socialbots, and the security implications on other software systems.

6.1 Users' Behavior

Given the results presented in Section 5, 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 hundreds of millions of user profiles, and thus, it is highly unlikely to have picked mostly socialbots.[13]

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 would increase the likelihood of accepting a friendship request, as discussed in Section 4.5.2. In fact, this resulted in the following correlation: the more mutual friends a socialbot had with a user, the higher the chance was that the accepted a friendship request send by the socialbot (Figure 5). The triadic closure, interestingly, operated from the users' side too, as the socialbots received a total of 331 friendship requests from their extended neighborhoods.

[13] Assuming that Facebook is mostly populated by genuine user profiles.

Second, the behavior depicted in Figure 4 matches the official statistics about users' login frequency in Facebook: 50% of the 750 million active Facebook users log on in any given day [20], and thus, it is expected that approximately half of the accepted friendship requests are observed within one day of the requests being sent by the socialbots.

Third and last, the users who were infiltrated during the bootstrapping phase, that is, those who were selected at random, showed another expected correlation in their behavior [64]: 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 Figures 6 and 7.

6.2 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—practically everything shared on Facebook by the infiltrated users—which could be used for large-scale user surveillance. Even though adversarial surveillance, such as online profiling [28], is a serious online privacy concern, we decided to only focus on users' data that have monetary value such as Personally Identifiable Information (PII).

After excluding all remaining friendships between the socialbots, 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.

For example, when considering all user profiles from *both* the direct and the extended neighborhoods of the socialbots, which added up to 1,088,840 profiles, we were able to collect $9,000$ Instant Messaging (IM) account IDs, $14,509$ physical mail addresses, $16,682$ phone numbers, $46,466$ email addresses, and $580,649$ birth dates, all after the infiltration.

6.3 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

Neighborhoods	Direct (%)		Extended (%)	
Profile Info	**Before**	**After**	**Before**	**After**
Gender	69.1	69.2	84.2	84.2
Birth Date	3.5	72.4	4.5	53.8
Married To	2.9	06.4	3.9	4.9
Worked At	2.8	4.0	2.8	3.2
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	0.9	19.0	0.7	1.3
Email Address	2.4	71.8	2.6	4.1
Phone Number	0.9	21.1	1.0	1.5
IM Account ID	0.6	10.9	0.5	0.8
Average	**13.3**	**34.9**	**15.4**	**23.7**

Table 3: Data revelation before and after operating the SbN, as percentage of neighborhoods size. (Direct neighborhoods = 3,055 profiles, Extended neighborhoods = 1,085,785 profiles)

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 constitute only 4% of the socialbots with 7% of the overall infiltration. Notice, however, that some of these socialbots got blocked by the FIS, which brings us to the following question: how did the FIS perform against the SbN we have operated?

After operating the SbN for eight weeks, only 20 profiles used by the socialbots were blocked by the FIS, and curiously, 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.[14] 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.

Finally, we noticed that using the commands `cluster` and `status_update`, as 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 opposed to 93% of the 15 profiles we used in our pilot study where

[14]Based on the content of a pop-up message that Facebook showed when we manually logged in using the blocked socialbots' profiles.

we decided not to use these commands. This, in a sense, reflects one of the drawbacks of relying on users' feedback.

6.4 Implications on Other Systems

This section explores the wider implications of large-scale infiltration in OSNs beyond today's social Web. In particular, we show that large-scale infiltration has alarming implications on software systems that use the social graph of OSNs to personalize, fine-tune, or bootstrap socially-informed services. We first outline the common assumption these systems make about the capabilities of adversaries in OSNs, and then, we focus on a case study in order to show that this assumption is generally unsafe and could lead to undesirable situations.

6.4.1 The Capability of a "Social Adversary"

Today, many software systems are socially informed due to the availability of huge, centralized OSNs that offer easy-to-use APIs. This enabled the development of a new set of socially-informed services that rely on the social graph of the used OSN. For example, OSNs are used to defend against Sybil attacks in peer-to-peer systems [76], enable knowledge-sharing and personalization in social search [31], model trust in recommender systems [71], and cooperate safely in online peer-based backup systems [65]. All of these systems, however, make the following assumption—implicitly or explicitly—about the capabilities of an adversary in OSNs:

> *It is particularly difficult for an adversary to establish arbitrarily many social connections between his OSN accounts (i.e., Sybils) and the rest of the users in the social graph.*

This assumption has the indirect effect of transforming the used OSN into a trust network, which brings in many advantages such as a well-defined trust metric, credibility, and hassle-free integration. In the following section, we evaluate how realistic this assumption is using a case study, and show that the outcomes of the studied system can drastically change once the assumption above does not hold in practice.

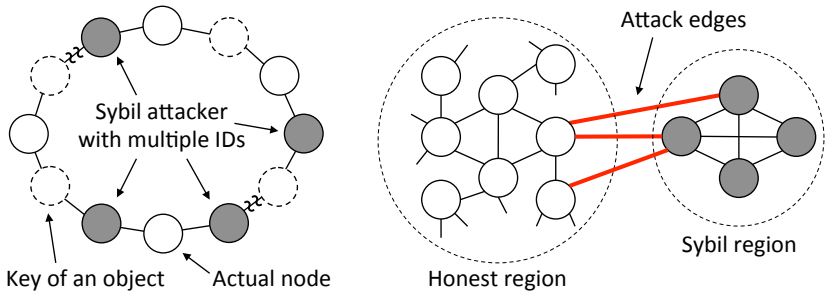
6.4.2 Case Study: Sybil Detection via Social Networks

As outlined in Section 2.2, the Sybil attack refers to the situation where an adversary controls a set of fake identities, each called a Sybil, and joins a targeted system multiple times under these Sybil identities.[15] The adversary can then mount many follow-up attacks using these Sybil identities in order to disrupt the targeted system. For example, an adversary can pollute the voting scheme of a trust and reputation system [35], subvert the routing and data replication services in a Distributed Hash Table (DHT) [68], or cripple many critical functions of a wireless sensor network such as routing, resource allocation, an misbehavior detection [52]. In this section, we focus on Sybil attacks in DHTs: decentralized, distributed systems that provide a lookup service, similar to a hash table, for a group of collaborating peers or nodes.

To defend against the Sybil attack, recent research propose to exploit the implicit social network among the nodes of a DHT in order to detect, and thereof limit, the number of Sybil nodes it may contain [15, 66, 67, 77, 78]. To demonstrate the main idea behind these Sybil defenses, let us consider the DHT in Figure 9(a). Each node is assigned a unique identifier and is responsible for maintaining a set of (key, value) pairs of shared objects, which are stored and maintained in the DHT. Any participating node can efficiently retrieve the value associated with a given key of an object by following the DHT protocol (e.g., Chord [60], Kademlia [43]). Figure 9(a) illustrate the case where an adversary join the DHT under four Sybil identities, which are represented as gray nodes.

A social network-based *Sybil detection protocol* is used to first learn the trust network among the peers in the DHT, and then use the graph structure of this social network to detect the Sybil nodes in the DHT. In particular, the protocol classifies the peers in the social network into two groups called *regions*: the Sybil region consisting of mostly Sybil nodes in the DHT, and the *honest* region consisting of mostly non-Sybil nodes, as shown in Figure 9(b). The edges in the social network connecting the Sybil region to the honest region are called the *attack edges*, and their quantity plays an important role in the accuracy of the classification. The basic idea is as follows: even if an adversary controls many Sybil nodes in the DHT, it is particularly difficult for him to establish many trust relationships with honest nodes in the social network, that is, it is hard to increase the number of attack edges in the network. Thus, one can

[15] The attack is named after the subject of the book Sybil by Flora Schreiber, a case study of a woman with multiple personality disorder whi had 16 different “alters”.



(a) Distributed Hash Table (b) Social Network

Figure 9: Detecting Sybil nodes in a typical DHT. In (a), dashed nodes represent keys of objects drawn from an identifier space, and un-dashed nodes represent peers where each peer has a unique identifier drawn from the same identifier space. In (b), an edge is added between two nodes if these nodes trust each other, forming a network of trust among the peers in the DHT. The structure of the resulting social network can be used to label each node either Sybil or honest (i.e., non-Sybil). This allows the classification of nodes in the DHT into two regions: the Sybil region and the honest region.

devise a detection mechanism that relies on this observation in order to group the nodes into honest and Sybil regions, and thus, cluster the network.

Most of the Sybil detection techniques via social networks are based on random walks and mixing times in a given social graph [15, 66, 67, 77, 78]. Yu [76], however, shows that if an adversary introduces few Sybil nodes such that the mixing time of the social graph is not affected, then no approach based on the mixing time can possibly tell that these nodes are Sybil. The *mixing time* describes how fast random walks in a given graph approach the stationary distribution of that graph, and is not affected as long as the adversary introduces at most as many Sybil nodes as the number of attack edges. Thus, if the mixing time is not effected by the introduction of Sybils, none of the major Sybil defenses will work effectively.[16] This result is not surprising as one would expect that if an adversary manages to establish many trust relationships with honest nodes, then the Sybil nodes will integrate into the honest region, rendering ineffective any clustering technique that is solely based on the social graph's structure. Clearly, to avoid this pitfall, Sybil defenses via social networks require the used network to have strong implicit trust such as friendship networks.[17] In online settings, however, the situation is different,

[16]The interested reader can refer to [76] for a formal treatment of this topic.

[17]The original proposal [77] requires the social network to be established "out-of-band" through real-life interactions, which is impractical in today's software systems.

as we show in the coming section.

6.4.3 Sybil Detection via OSNs

Today's Internet-based, open-access software systems, either centralized or distributed, allow a user to join the system by presenting an online identity that is borrowed from an existing OSN, which is usually achieved by utilizing a Web single sign-on technology such as Facebook Connect [37]. We showed, however, that OSNs such as Facebook are highly vulnerable to large-scale infiltration, where a network of *few* socialbots can be used to establish *arbitrarily many* friendships with Facebook users. Therefore, and according to our discussion in Section 6.4.2, we do not recommend using an OSN such as Facebook to detect Sybil nodes in open-access, online systems (e.g., peer-to-peer file sharing networks), as they are expected to be ineffective given the infiltration capability of adversaries in OSNs.

The above conclusion extends to all open-access systems that integrate OSNs such as Facebook, especially those that are Web-based. For example, Facebook reports that more than 500 million of its users interact with third-party applications on its Web platform or experience Facebook platform on other websites every month, where more than seven million applications and websites are integrated with Facebook [20]. Apparently, this is a welcoming haven for Sybil attackers in today's social Web, including SbN botherders and their affiliates. Defending against threats such as large-scale infiltration is thus important, which is a topic of future research.

7 Conclusion and Future Work

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.

To that end, we are currently investigating two directions in the defense side. The first involves understanding why users tend to accept friendship requests from stranger in general, which could inform the design of a user-centric security control that allows users to kmake more informed decisions. The second, on the other hand, involves characterizing social network-based Sybil defenses under the assumption of a social adversary, which could provide us with insights on how to design similar defenses that are resilient to such a resourceful adversary.

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) and GRAND Network of Centers of Excellence (NCE).

References

[1] Facebook Open Graph Protocol. `http://developers.facebook.com/docs/opengraph`. 13, 21

[2] Sick profile maker. `http://sickmarketing.com/sick-profile-maker`. 11

[3] Jet bots. `http://allbots.info`, 2011. 6

[4] R. J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley Publishing, 2 edition, 2008. ISBN 9780470068526. 9

[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. 5, 9, 10, 18

[6] J. Bates. Sniffing out socialbots: The combustive potential of social media-based algorithms. `http://www.businessinsider.com/sniffing-out-socialbots-the-combustive-potential-of-social-media-based-algorithms-2011-12`, December 2011. 5

[7] BBC News. Socialbots used by researchers to "steal" facebook data. http://www.bbc.co.uk/news/technology-15553192, November 2011. 21

[8] 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. ISBN 978-1-60558-487-4. doi: http://doi.acm.org/10.1145/1526709.1526784. 9, 10, 12, 20

[9] J. Bollen, H. Mao, and X. Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1 – 8, 2011. ISSN 1877-7503. doi: 10.1016/j.jocs.2010.12.007. URL http://www.sciencedirect.com/science/article/pii/S187775031100007X. 5

[10] 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. ISBN 978-1-60558-247-4. doi: http://doi.acm.org/10.1145/1520340.1520402. 20

[11] D. Boyd. Social media is here to stay... Now what? *Microsoft Research Tech Fest*, February 2009. 5

[12] M. Broersma. idefense: 1.5 million facebook accounts for sale. http://www.zdnet.co.uk/news/security-threats/2010/04/23/idefense-15-million-facebook-accounts-for-sale-40088751/, April 2010. 11

[13] 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. ISBN 978-1-60558-007-4. doi: http://doi.acm.org/10.1145/1460563.1460628. 9

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

[15] G. Danezis and P. Mittal. SybilInfer: Detecting sybil nodes using social networks. In *NDSS*, Feb. 2009. 30, 31

[16] 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. ISBN 3-540-44179-4. 8

[17] DSLReports. Trojan horse, and virus FAQ. http://www.broadbandreports.com/faq/trojans/1.0_Trojan_horses, July 2011. 21

[18] 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. ISBN 978-1-60558-639-7. doi: http://doi.acm.org/10.1145/1774088.1774483. 10, 20

[19] 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. ISSN 10836101. doi: 10.1111/j.1083-6101.2007.00367.x. URL http://dx.doi.org/10.1111/j.1083-6101.2007.00367.x. 20

[20] Facebook. Facebook Statistics. http://www.facebook.com/press, March 2011. 18, 20, 27, 32

[21] Facebook. Facebook security. http://www.facebook.com/security, July 2011. 21

[22] Facebook Devil. Facebook devil: Facebook profile creator. http://www.facebookdevil.com/, January 2011. 11

[23] N. FitzGerald. New Facebook worm. http://fitzgerald.blog.avg.com/2009/11/, November 2009. 12

[24] T. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(3Ű4):143–166, 2003. 10

[25] 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. 23

[26] 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. ISBN 978-1-4503-0245-6. doi: http://doi.acm.org/10.1145/1866307.1866311. URL http://doi.acm.org/10.1145/1866307.1866311. 6

[27] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon. Peer-to-peer botnets: overview and case study. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 1–1, Berkeley, CA, USA, 2007. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1323128.1323129. 23

[28] R. Gross, A. Acquisti, and H. J. Heinz, III. Information revelation and privacy in online social networks. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80, New York, NY, USA, 2005. ACM. ISBN 1-59593-228-3. doi: http://doi.acm.org/10.1145/1102199.1102214. 27

[29] 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. 9, 11

[30] 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. URL http://dx.doi.org/10.1007/978-3-642-05437-2_2. 10

[31] D. Horowitz and S. D. Kamvar. The anatomy of a large-scale social search engine. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 431–440, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: http://doi.acm.org/10.1145/1772690.1772735. URL `http://doi.acm.org/10.1145/1772690.1772735`. 29

[32] M. Huber, S. Kowalski, M. Nohlberg, and S. Tjoa. Towards automating social engineering using social networking sites. *Computational Science and Engineering, IEEE International Conference on*, 3:117–124, 2009. doi: http://doi.ieeecomputersociety.org/10.1109/CSE.2009.205. 9

[33] 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. 13, 20

[34] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Commun. ACM*, 50(10):94–100, 2007. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/1290958.1290968. 9, 20

[35] A. Jøsang and J. Golbeck. Challenges for robust of trust and reputation systems, Sept. 2009. 30

[36] 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. ISSN 0007-6813. doi: DOI:10.1016/j.bushor.2009.09.003. URL `http://www.sciencedirect.com/science/article/pii/S0007681309001232`. 5

[37] M. N. Ko, G. Cheek, M. Shehab, and R. Sandhu. Social-networks connect services. *Computer*, 43(8):37 –43, aug. 2010. ISSN 0018-9162. doi: 10.1109/MC.2010.239. 32

[38] C. Lampe, N. Ellison, and C. Steinfield. A face(book) in the crowd: social searching vs. social browsing. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, CSCW '06, pages 167–170, New York, NY, USA, 2006. ACM. ISBN 1-59593-249-6. doi: http://doi.acm.org/10.1145/1180875.1180901. URL `http://doi.acm.org/10.1145/1180875.1180901`. 20

[39] C. Lampe, N. B. Ellison, and C. Steinfield. Changes in use and perception of facebook. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, CSCW '08, pages 721–730, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-007-4. doi: http://doi.acm.org/10.1145/1460563.1460675. URL `http://doi.acm.org/10.1145/1460563.1460675`. 20

[40] 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. ISBN 978-1-60558-085-2. doi: http://doi.acm.org/10.1145/1367497.1367620. URL `http://doi.acm.org/10.1145/1367497.1367620`. 18

[41] G. Livingston. Social media: The new battleground for politics. http://mashable.com/2010/09/23/congress-battle-social-media/, September 2010. 5

[42] D. Lowd and C. Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 641–647, New York, NY, USA, 2005. ACM. ISBN 1-59593-135-X. doi: http://doi.acm.org/10.1145/1081870.1081950. URL http://doi.acm.org/10.1145/1081870.1081950. 23

[43] P. Maymounkov and D. Maziéres. Kademlia: A peer-to-peer information system based on the xor metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS'01, pages 53–65, London, UK, 2002. Springer-Verlag. 30

[44] 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. 5

[45] 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. ISBN 978-1-59593-908-1. doi: http://doi.acm.org/10.1145/1298306.1298311. 12

[46] 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. 5

[47] 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. ISBN 888-7-6666-5555-4. URL http://portal.acm.org/citation.cfm?id=1929820.1929858. 10

[48] M. Motoyama, D. McCoy, K. Levchenko, S. Savage, and G. M. Voelker. Dirty jobs: The role of freelance labor in web service abuse. In *Proceedings of the 20th USENIX Security Symposium*, Aug. 2011. URL http://www.usenix.org/events/sec11/tech/full_papers/Motoyama.pdf. 11

[49] 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. 20

[50] 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. ISBN 978-0-7695-3689-7. doi: 10.1109/ASONAM.2009.61. URL http://portal.acm.org/citation.cfm?id=1602240.1602706. 9, 19

[51] M. Nanis, I. Pearce, and T. Hwang. Pacific social architecting corporation: Field test report. http://pacsocial.com/, November 2011. 10

[52] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis defenses. In *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*, pages 259 – 268, april 2004. doi: 10.1109/IPSN.2004.1307346. 30

[53] S. Patil. Social network attacks surge. http://www.symantec.com/connect/blogs/social-network-attacks-surge, June 2011. 6

[54] 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. 6

[55] A. Rapoport. Spread of information through a population with socio-structural bias: I. assumption of transitivity. *Bulletin of Mathematical Biology*, 15:523–533, 1953. ISSN 0092-8240. URL http://dx.doi.org/10.1007/BF02476440. 10.1007/BF02476440. 6, 18

[56] 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. ISBN 978-1-4503-0637-9. doi: http://doi.acm.org/10.1145/1963192.1963301. URL http://doi.acm.org/10.1145/1963192.1963301. 5

[57] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. ISBN 0387212396. 22

[58] F. Salem and R. Mourtada. Civil movements: The impact of Facebook and Twitter. *The Arab Social Media Report*, 1(2), 2011. URL http://www.dsg.ae/portals/0/ASMR2.pdf. 5

[59] 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. ISBN 978-1-4503-0728-4. doi: http://doi.acm.org/10.1145/1989656.1989664. URL http://doi.acm.org/10.1145/1989656.1989664. 6, 20

[60] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking (TON)*, 11(1):17–32, 2003. ISSN 1063-6692. doi: http://dx.doi.org/10.1109/TNET.2002.808407. 30

[61] 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. ISBN 978-1-4503-0133-6. doi: http://doi.acm.org/10.1145/1920261.1920263. URL http://doi.acm.org/10.1145/1920261.1920263. 6

[62] C. Taylor. Why not call it a Facebook revolution? http://edition.cnn.com/2011/TECH/social.media/02/24/facebook.revolution/, February 2011. 5

[63] The Web Ecology Project. The 2011 socialbots competition. http://www.webecologyproject.org/category/competition/, January 2011. 10

[64] 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. ISSN 1083-6101. doi: 10.1111/j.1083-6101.2008.00409.x. URL http://dx.doi.org/10.1111/j.1083-6101.2008.00409.x. 18, 27

[65] D. N. Tran, F. Chiang, and J. Li. Friendstore: cooperative online backup using trusted nodes. In *Proceedings of the 1st Workshop on Social Network Systems*, SocialNets '08, pages 37–42, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-124-8. doi: http://doi.acm.org/10.1145/1435497.1435504. URL http://doi.acm.org/10.1145/1435497.1435504. 29

[66] N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, NSDI'09, pages 15–28, Berkeley, CA, USA, 2009. USENIX Association. 30, 31

[67] N. Tran, J. Li, L. Subramanian, and S. Chow. Optimal sybil-resilient node admission control. In *INFOCOM, 2011 Proceedings IEEE*, pages 3218 –3226, april 2011. 30, 31

[68] G. Urdaneta, G. Pierre, and M. van Steen. A survey of DHT security techniques. *ACM Computing Surveys*, 43(2), Jan. 2011. http://www.globule.org/publi/SDST_acmcs2009.html. 30

[69] 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. 5

[70] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard AI problems for security. In *EUROCRYPT*, pages 294–311, 2003. 10

[71] 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. ISSN 1387-2532. doi: 10.1007/s10458-007-9021-x. URL http://dx.doi.org/10.1007/s10458-007-9021-x. 29

[72] J. Weizenbaum. Eliza — a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9:36–45, January 1966. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/365153.365168. URL http://doi.acm.org/10.1145/365153.365168. 16

[73] G. Yan, G. Chen, S. Eidenbenz, and N. Li. Malware propagation in online social networks: nature, dynamics, and defense implications. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, pages 196–206, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0564-8. doi: http://doi.acm.org/10.1145/1966913.1966939. URL http://doi.acm.org/10.1145/1966913.1966939. 5

[74] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai. Uncovering social network sybils in the wild. In *IMC*, Berlin, Germany, November 2011. 6, 18

[75] H. Yeend. Breaking CAPTCHA without OCR. http://www.puremango.co.uk/2005/11/breaking_captcha_115/, November 2005. URL http://www.puremango.co.uk/2005/11/breaking_captcha_115/. 10

[76] H. Yu. Sybil defenses via social networks: a tutorial and survey. *SIGACT News*, 42: 80–101, October 2011. ISSN 0163-5700. 29, 31

[77] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. *SIGCOMM Comput. Commun. Rev.*, 36:267–278, August 2006. ISSN 0146-4833. 30, 31

[78] 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. ISBN 978-0-7695-3168-7. doi: 10.1109/SP.2008.13. URL http://portal.acm.org/citation.cfm?id=1397759.1398053. 30, 31