Security Analysis of Malicious Socialbots on the Web

by

Yazan Boshmaf

B. Computer Engineering, Jordan University of Science and Technology, 2005
M. Information Technology, University of Stuttgart, 2008

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Electrical and Computer Engineering)

The University of British Columbia
(Vancouver)

May 2015

Abstract

The open nature of the Web, online social networks (OSNs) in particular, makes it possible to design *socialbots*—automation software that controls fake accounts in a target OSN, and has the ability to perform basic activities similar to those of real users. In the wrong hands, socialbots can be used to infiltrate online communities, build up trust over time, and then engage in various malicious activities.

This dissertation presents an in-depth security analysis of malicious socialbots on the Web, OSNs in particular. The analysis focuses on two main goals: (1) to characterize and analyze the vulnerability of OSNs to cyber attacks by malicious socialbots, *social infiltration* in particular, and (2) to design and evaluate a countermeasure to efficiently and effectively defend against socialbots.

To achieve these goals, we first studied social infiltration as an organized campaign operated by a *socialbot network* (SbN)—a group of programmable socialbots that are coordinated by an attacker in a botnet-like fashion. We implemented a prototypical SbN consisting of 100 socialbots and operated it on Facebook for 8 weeks. Among various findings, we observed that some users are more likely to become victims than others, depending on factors related to their social structure. Moreover, we found that traditional OSN defenses are not effective at identifying automated fake accounts or their social infiltration campaigns.

Based on these findings, we designed Íntegro—an *infiltration-resilient defense system* that helps OSNs detect automated fake accounts via a user ranking scheme. In particular, Íntegro relies on a novel approach that leverages *victim classification* for robust graph-based fake account detection, with provable security guarantees.

We implemented Íntegro on top of widely-used, open-source distributed systems, in which it scaled nearly linearly. We evaluated Íntegro against SybilRank—the state-of-the-art in graph-based fake account detection—using real-world datasets and a large-scale, production-class deployment at Tuenti, the largest OSN in Spain with more than 15 million users. We showed that Íntegro significantly outperforms SybilRank in ranking quality, allowing Tuenti to detect at least 10 times more fake accounts than their current abuse detection system.

Preface

"Акъылым уасэ иІэкъым, гъэсэныгъэм гъунэ иІэкъым"
"Intellect is priceless, education has no limit" — Circassian proverb [59]

This research was the product of a fruitful collaboration between the author of the dissertation and the following people: Ildar Muslukhov, Konstantin Beznosov (co-advisor), and Matei Ripeanu (co-advisor) from the University of British Columbia, Dionysios Logothetis and Georgos Siganos from Telefónica Research, and Jorge Rodriguez Lería and Jose Lorenzo from Tuenti, Telefónica Digital.

It is worth mentioning that the work presented herein consists of research studies that have been published or under review in peer-reviewed international conferences, workshops, and journals. In particular, the characterization study presented in Chapter 2, and partly discussed in Chapter 4, led to the following publications:

- Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu. The Socialbot Network: When Bots Socialize for Fame and Money. In *Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC '11)*, pp. 93–102, Orlando, FL, USA, 2011 (best paper award, 20% acceptance rate).

- Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu. Key Challenges in Defending against Malicious Socialbots. In *Proceedings of the 5th Annual USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '12)*, San Jose, CA, USA, 2012 (18% acceptance rate).

- Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu. Design and Analysis of a Social Botnet. *Computer Networks*, Elsevier, 57(2), pp. 556–578, 2013 (1.87 impact factor).

The results related to the system design part, which are reported in Chapter 3, are presented in the following refereed publications:

- Y. Boshmaf, K. Beznosov, and M. Ripeanu. Graph-based Sybil Detection in Social and Information Systems. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '13)*, pp. 466–473, Niagara Falls, ON, Canada, 2013 (best paper award, 13% acceptance rate).

- Y. Boshmaf, D. Logothetis, G. Siganos, J. R. Lería, J. Lorenzo, M. Ripeanu, and K. Beznosov. Íntegro: Leveraging Victim Prediction for Robust Fake Account Detection in OSNs. In *Proceedings of the 2015 Annual Network and Distributed System Security Symposium (NDSS '15)*, San Diego, CA, USA, 2015 (16% acceptance rate).

- Y. Boshmaf, D. Logothetis, G. Siganos, J. R. Lería, J. Lorenzo, M. Ripeanu, and K. Beznosov. Thwarting Fake Accounts in Online Social Networks by Predicting their Victims. Under review (submitted in March, 2015).

The discussion in Chapter 4 is partially influenced by ideas and findings which led to publications at the following refereed workshops and conferences:

- S-T. Sun, Y. Boshmaf, K. Hawkey, and K. Beznosov. A Billion Keys, but Few Locks: The Crisis of Web Single Sign-On. In *Proceedings of the 2010 Workshop on New Security Paradigms (NSPW '10)*, Concord, MA, USA, 2010 (32% acceptance rate).

- H. Rashtian, Y. Boshmaf, P. Jaferian, and K. Beznosov. To Befriend or Not? A Model for Friend Request Acceptance on Facebook. In *Proceedings of the 11th Symposium On Usable Privacy and Security (SOUPS '14)*, Menlo Park, CA, USA, 2014 (27% acceptance rate).

Table of Contents

List of Tables

List of Figures

Acknowledgments

First and foremost, I would like to thank my kind advisors, Konstantin Beznosov and Matei Ripeanu, for giving me the opportunity to venture into different topics and disciplines, and for patiently guiding me through this journey.

Second, this work would have been no fun without my fabulous collaborators and supportive friends. In no particular order, I give my special thanks to you all: Dionysios Logothetis, Georgos Siganos, Jorge Rodriguez Lería, Jose Lorenzo, Ildar Muslukhov, Pooya Jaferian, San-Tsai Sun, Hootan Rashtian, Primal Wijesekera, Ivan Cherapau, Tareq AlKhatib, Bader AlAhmad, Hussam Ashab, Antone Dabeet, Daniel Lopez, Nima Fartash, Mohammad Shamma, and Alex Dauphin.

Third, I would like to thank all members of LERSSE and NetSysLab for their feedback and constructive discussions. I am greatly thankful to Tim Hwang from Pacific Social, Chris Sumner from the Online Privacy Foundation, Ulf Lindqvist from SRI International, Hasan Cavusoglu from Sauder School of Business, Joseph McCarthy from University of Washington–Bothell, the Alexander von Humboldt Foundation, and the Natural Sciences and Engineering Research Council (NSERC) of Canada for their external support and feedback during my studies. I am vastly indebted to the University of British Columbia for a generous doctoral fellowship.

Fourth, salutes to my colleagues at Facebook, Telefónica, Microsoft, and Sophos for enriching internship experiences: Kyle Zeeuwen, David Cornell, Dmitry Samosseiko, Yuchun Tang, Matt Jones, Nikolaos Laoutaris, and Mihai Budiu.

Last but not least, I would like to thank my family. Words cannot express how thankful I am for your constant support and love over these years.

Dedication

To my parents who never stopped believing in me

Chapter 1

Introduction

Our personal and professional lives have gone digital. We live, work, and play in the cyber-space. We use the Web every day to talk, email, text, and socialize with family, friends, and colleagues. Among a plethora of social Web services, online social networks (OSNs), such as Facebook[1] and Twitter[2], have had the highest popularity and impact on the way we engage with the Web on a daily basis [11].

With more than one billion active users [30, 119], OSNs have attracted third parties who exploit them as effective online social media to reach out to and potentially influence a large and diverse population of web users [24, 62]. For example, OSNs were heavily employed by Obama's 2012 campaign team who raised $690 million online, up from $500 million in 2008 [100]. In addition, it has been argued that OSNs were one of the major enablers of the recent Arab Spring in the Middle East [89]. This pervasive integration of OSNs into everyday life is rapidly becoming the norm, and arguably, it is here to stay. Today's social Web, however, is not exclusive to only human beings. In fact, online personas and their social interactions can be programmed and fully automated. As Hwang et al. put it [56]: "digitization drives botification; the use of technology in a realm of human activity enables the creation of software to act in lieu of human."

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

In this dissertation, we analyze the threat of malicious *socialbots*—automated personas that infiltrate OSNs like virtual con men, build up trust over time, and then exploit it to promote personalized messages for adversarial objectives.[3]

1.1 Problem Overview

In what follows, we first define the threat of malicious socialbots (Section 1.1.1), explain why it is important to defend against (Section 1.1.2), discuss who is legally liable in case of harm or damage (Section 1.1.3), and then elaborate why mitigating this threat is a hard socio-technical problem (Section 1.1.4).

1.1.1 What is the Threat?

A new breed of computer programs called socialbots are emerging, and they can influence users online [79]. A *socialbot* is an automation software that controls a fake account in a target OSN, and has the ability to perform basic activities similar to those of real users, such as posting messages and sending friend requests.

What makes a socialbot different from *self-declared bots*, such as Twitter bots that post weather forecasts, or *spambots*, which are bots that massively distribute unsolicited messages to non-consenting users, is that *it is designed 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 [56].

In the wrong hands, a socialbot can be used to *infiltrate* users in a target OSN and reach an influential position, that is, to adversely manipulate the *social graph* of the OSN by connecting with a large number of its users. We discuss the threat model is more detail in Sections 2.3.2 and 3.1.1

[3] Use of the plural pronoun is customary even in solely authored research work. However, given the subject of this dissertation, its use herein is particularly ironic, following Douceur's seminal paper [22] on multiple-identity or *Sybil attacks* in distributed systems.

1.1.2 What is at Stake?

First of all, a malicious socialbot can pollute the target OSN with a large number of non-genuine social relationships. This means that it is unsafe to treat the infiltrated OSN as a network of trust anymore, which goes against the long-term health of the OSN ecosystem [11]. Put differently, third-party applications and websites would have to identify and remove fake user accounts before integrating with such OSNs, which has been shown to be a non-trivial task [137].

Second, once a socialbot infiltrates the target OSN, it can exploit its new position in the network to spread misinformation and bias the public opinion [113], perform online surveillance [95], and even influence algorithmic trading [8]. For example, Ratkiewicz et al. [98] described the use of Twitter bots to run astroturf campaigns during the 2010 U.S. midterm elections. Moreover, a socialbot can exploit its position in the network to distribute malicious content, such as malware and botnet executables [132]. For example, the Koobface botnet [112] propagates by hijacking user accounts of infected machines, after which it uses these accounts to send messages containing a malicious link to other OSN users. When clicked, the link points to a compromised website that attempts to infect its visitors with the Koobface malware using drive-by download attacks [132].

Third and last, as a socialbot infiltrates the target OSN, it can harvest private user data such as email addresses, phone numbers, and other personally identifiable information that have monetary value. For an attacker, such data are valuable and can be used for online profiling and large-scale email spam and phishing campaigns [58]. It is thus not surprising that similar socialbots are being offered for sale at underground markets, along with other commodities for OSN abuse [114].

1.1.3 Who is Liable?

It is often the case that OSN users or third parties are liable for any damage caused by OSN platforms or services, not OSN operators [78]. For example, Facebook's Terms of Service explicitly states under §15 entitled "Disputes" that [33]:

> "We [Facebook and subsidiaries] try to keep Facebook up, bug-free, and safe, but you [the end user] use it at your own risk. We are providing Facebook as is without any express or implied warranties. [...] We do not guarantee that Facebook will always be safe, secure or error-free. [...] Facebook is not responsible for the actions, content, information, or data of third parties, and you release us, our directors, officers, employees, and agents from any claims and damages, known and unknown, arising out of or in any way connected with any claim you have against any such third parties."

From privacy law perspective, the American courts, for example, lack coherent and consistent methodology for determining if an individual has a reasonable expectation of privacy in a particular fact that has been shared with one or more persons in social networks [106]. In online settings, this methodological weakness can have severe implications, as OSNs like Facebook end up facilitating peer-to-peer privacy violations in which users harm each others' privacy interests [43]. While policymakers cannot make such OSNs completely safe, they can help people use it safely by introducing policy interventions, such as a strengthened public-disclosure tort and a right to opt out completely [43]. It is also in the interest of OSN operators to keep their platforms as safe and secure as possible by preventing or detecting malicious activities, including automated fake accounts [17, 33]. This commitment is evident by the large investments made by OSNs to hire teams of security engineers in order to develop, deploy, and maintain sophisticated defense mechanisms [15, 104, 135].

1.1.4 Why is the Threat Hard to Mitigate?

Recently, a number of *feature-based detection* techniques were proposed to automatically identify OSN bots, or automated fake accounts, based on their abnormal behavior [108, 127, 135]. By extracting *features* that describe certain user behaviors from recent activities (e.g., frequency of friend requests), a fake account classifier is trained using various machine learning techniques. For example, Stein et

al. [104] presented Facebook's "immune system"—an adversarial machine learning system that performs real-time checks and classifications on every read and write action on Facebook's database, which are based on features extracted from user accounts and their activities. Accordingly, it is expected that such an adversarial learning system to be effective at identifying and blocking spambots. Socialbots, however, are more deceptive than spambots as they are designed to behave "normally" by posing as average OSN users [56]. Armed with today's artificial intelligence advancements, it is feasible to orchestrate a network of socialbots that sense, think, and act cooperatively just like human beings. An attacker, for example, can use *adversarial classifier reverse engineering* techniques [74] in order to learn sufficient information about the employed security defenses and then evade detection by constructing an adversarial strategy that minimizes the chance of a socialbot being classified as abnormal or fake, sometimes down to zero.

Graph-based detection techniques [2, 123, 137], as an alternative to feature-based detection, analyze the social graph in order to partition it into two *regions* separating real accounts from fakes. Such techniques, however, are expected to be less effective at identifying malicious socialbot, as they make strong assumptions about the capabilities of online attackers that do not hold in practice. For example, it is often assumed that the regions are sparsely connected, that is, automated fake accounts cannot establish many social relationships with real users, as this would require exceptional social engineering [137]. This assumption does not hold when malicious socialbots are used, because each bot is expected to gradually infiltrate and integrate into the online community it targets, resembling the scenario when a new user joins an OSN and starts connecting with others [56].

To this end, detecting malicious socialbots in OSNs is expected to result in an arms race, which will keep both the defenders and the attackers busy, depending on their available resources. The threat of socialbots could be mitigated by eliminating the factors that make their automated operation feasible in the first place. Doing so, however, involves solving a number of hard socio-technical challenges, which we identify later in Section 4.1. Alternatively, the OSN can facilitate strict

account admission policies that limit the service offered to newly registered users. As we discussed in Section 4.2, this can negatively impact user experience.

1.2 Goals and Methodology

Our ultimate goal is twofold: (1) to understand and characterize what makes OSNs vulnerable to cyber-attacks by malicious socialbots, to social infiltration in particular, and (2) to design a new countermeasure to effectively and efficiently defend against malicious socialbots. In order to achieve these goals, we divided our investigation into two parts: *Threat characterization*, which is presented in Chapter 2, and *countermeasure design*, which is presented in Chapter 3.

As for our research methodology, we followed the standard iterative process in improving computer security, which starts with vulnerability analysis, followed by security requirements specification and countermeasure design, and then ends with security assurance [4]. We used quantitative and analytical research methods to characterize social infiltration in OSNs and design a robust defense mechanism against it with provable security guarantees. These methods included controlled experiments, statistical analysis, simulations, mathematical modeling and analysis, and real-world system deployment. We provide more detailed description of our research methodology in Sections 2.4.2 and 3.4.2.

1.3 Research Summary

While the motivations for operating socialbots and the technical mechanisms that enable them remain rich areas of research, we focus our investigation around four research questions (RQs) covering the threat characterization part (Section 1.3.1) and the countermeasure design part (Section 1.3.2). Figure 1.1 summarizes the research questions and contributions of this dissertation.

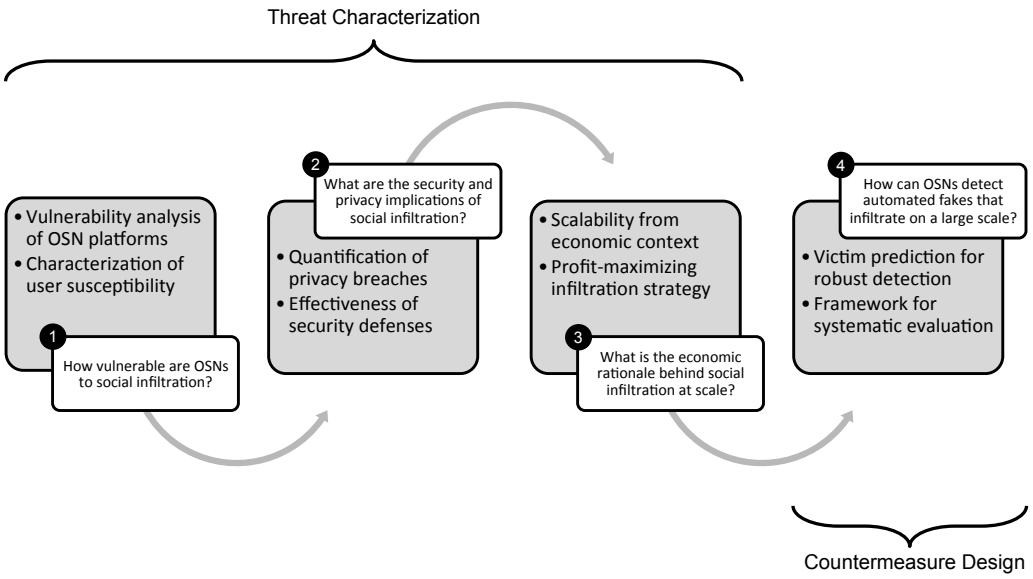


Figure 1.1: Research questions and contributions

1.3.1 Threat Characterization

In Chapter 2, we consider profit-driven attackers who infiltrate OSNs using malicious socialbots. In particular, the following questions guide our *characterization of social infiltration* in OSNs:

- **RQ1:** How vulnerable are OSNs to social infiltration?
- **RQ2:** What are the security and privacy implications of social infiltration?
- **RQ3:** What is the economic rationale behind social infiltration at scale?

To address these questions, we studied social infiltration as an organized campaign run by a network of socialbots. In particular, we adopted the design of web-based botnets [101] and defined what we call a *socialbot network* (SbN): A group of programmable socialbots that are coordinated by an attacker using a software controller called the *botmaster*. In our design, the botmaster follows simple infiltration strategies that exploit known social behaviors of users in OSNs in order to increase the scale of infiltration and its success rate.

Main Findings

We implemented an SbN consisting of 100 socialbots and a single botmaster. We operated this SbN on Facebook for 8 weeks in early 2011. During that time, the socialbots sent a total of 9,646 friend requests to *attacked users*, out of which 3,439 requests were accepted by *victims*. This resulted in an average success rate of 35.7%. We recorded all data related to user behavior, along with all accessible users' profile information.

The main findings of this part of the dissertation are the following:

Finding 1 (under RQ1)**:** *OSNs such as Facebook suffer from inherent vulnerabilities that enable an attacker to automate social infiltration on a large scale*

We analyzed the vulnerability of Facebook to malicious automation. We found that OSNs such as Facebook employ ineffective CAPTCHAs, allow multiple accounts to be created by the same user, hide the social graph but permit any user to crawl it, and provide social APIs and web platforms that are relatively easy to exploit or reverse engineer. Along with poorly designed end-user privacy controls, these vulnerabilities represent the *enabling factors* that make operating socialbots feasible in the first place.

Finding 2 (under RQ1)**:** *Some users are more likely to become victims than others, which partly depends on factors related to their social structure.*

We found that OSN users are not careful when accepting connection requests, especially 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%, where the bots shared at least 10 mutual friends with the victims. Moreover, we found that the more friends a user has, the more likely the user is to accept friend requests sent by fakes posing as strangers, regardless to their gender or number of mutual friends.

Finding 3 (under RQ2)**:** *An SbN can seriously breach the privacy of users, where personally identifiable information is compromised.*

Subject to user privacy settings, we showed that an attacker can gain access to 2.6 times more private user data by running a social infiltration campaign. These data include birth dates, email addresses, phone numbers, and other personally identifiable information. This privacy breach includes the private data of "friends of friends," that is, users who have not been infiltrated by socialbots but are friends with their victims.

Finding 4 (under RQ2)**:** *Traditional OSN defenses are not effective at identifying automated fake accounts nor their social infiltration campaigns.*

Even though Facebook uses a sophisticated feature-based abuse detection system that relies on various machine learning techniques to identify fake accounts [104], we found that only 20% of the fakes controlled by socialbots were correctly identified and suspended. However, these suspended accounts were in fact manually flagged by concerned users, rather than by Facebook. In addition, while the average Facebook user had approximately 100 friends [39], we found that 50% of the socialbots infiltrated more than 35 victims and up to 120, which means fakes are not always loosely connected to real accounts in the social graph. This, as a result, renders graph-based fake account detection systems, which aim to find sparse cuts to separate real accounts from fakes, ineffective in practice.

Finding 5 (under RQ3)**:** *For an SbN to be scalable in terms of number of attacked users, it ought to have a fixed size in terms of number of socialbots.*

We developed an economic model for a profit-driven social infiltration. Using cost-volume-profit (CVP) analysis, we found that when operating an SbN using a growing number of socialbots, there is no benefit—if not a loss—from scaling the SbN and attacking even more users. Due to linear cost dependance, the attacker makes the same profit, at best, if 1K or 100K users are attacked, for example.

Finding 6 (under RQ3)**:** *Operating an SbN at scale is expected to be profitable, but it is not particularly attractive as an independent business.*

We derived two business models for social infiltration under which an attacker can make non-zero profit. In the first model, the attacker sells harvested user data through infiltrating as many users as possible, following a scalable *data-driven* infiltration strategy. In the second model, the attacker receives a lump-sum payment through infiltrating a predefined number of users, following a non-scalable *target-driven* infiltration strategy. However, due to poor market incentives, our analysis indicates that scalable social infiltration is not sustainable as an independent business, and a rational botherder would utilize an SbN as a *monetization platform* for more profitable underground commodities.

Contributions

In summary, this part of the dissertation makes the following main contributions:

Contribution 1: *Demonstrating the feasibility of social infiltration in OSNs.*

We performed the first comprehensive study that shows the feasibility of social infiltration in OSNs such as Facebook. In particular, we designed and evaluated a social botnet on Facebook, which resulted in new or confirmatory findings related to platform vulnerabilities, user susceptibility to social infiltration, data breaches, and the effectiveness of fake account detection mechanisms against malicious socialbots that are capable of social infiltration at scale (Findings 1–4).

Contribution 2: *Economic analysis of social infiltration at scale.*

We developed a mathematical model to analyze the scalability of social infiltration from an economic context. We used this model to derive profit-maximizing infiltration strategies that satisfy different scalability requirements. We also evaluated the economic feasibility of social infiltration under these strategies using data collected from Facebook, which resulted in new findings related to the dynamics of social infiltration as a monetization platform for OSN abuse (Findings 5–6).

1.3.2 Countermeasure Design

In Chapter 3, we consider attackers who can run a social infiltration campaign at a large scale using a set of automated fake accounts, or socialbots. Specifically, each fake account can perform social activities similar to those of real users, including befriending other real users. Motivated by Finding 4, our design of an *infiltration-resilient fake account detection* mechanism tackles the following question:

- **RQ4:** How can OSNs detect automated fakes that infiltrate on a large scale?

To address this question, we designed Íntegro—a robust and scalable defense system that helps OSNs detect automated fake accounts via a user ranking scheme.[4] The system is suitable for OSNs whose users declare bidirectional social relationships (e.g., Tuenti,[5] RenRen,[6] LinkedIn,[7] Facebook), with the ranking process being completely transparent to users. While the ranking scheme is graph-based, the graph is preprocessed first and annotated with information derived from feature-based detection techniques, similar to those employed by Facebook's immune system. This approach of integrating user-level activities into graph-level structures positions Íntegro as the first feature-and-graph-based detection mechanism.

Design Overview

Our design directly follows from Finding 2 and is based on the observation that *victims*—real accounts whose users have accepted friend requests sent by fakes—are useful for designing robust fake account detection mechanisms. In particular, Íntegro uses simple account features (e.g., gender, number of friends, time since last update), which are cheap to extract from user-level activities, to train a victim classifier in order to identify *potential victims* in the OSNs. As attackers have no control over victim accounts nor their activities, a victim classifier is inherently

[4] In Spanish, the word "íntegro" means integrated, which suites our novel approach of integrating user-level activities into graph-level structures.

[5] http://tuenti.com

[6] http://renren.com

[7] http://linkedin.com

more resilient to adversarial attacks than a similarly-trained fake account classifier. Moreover, as victims are directly connected to fakes in the graph, they represent a natural "borderline" that separates real accounts from fakes.

Íntegro makes use of this observation by consistently assigning lower weights to edges incident to potential victims than other accounts, after which it ranks user accounts based on the landing probability of a short, supervised random walk that starts from a known real account. The random walk is "short," as it is terminated early before it converges. The walk is "supervised," as it is biased towards traversing nodes that are reachable through higher-weight paths. Therefore, this modified random walk is likely to stay within the subgraph consisting of real accounts, and thus *most real accounts receive higher ranks than fakes*. Unlike SybilRank [15], which is the state-of-the-art in graph-based fake account detection, we do not assume sparse connectivity between real and fake accounts. This makes Íntegro the first fake account detection system that is *robust against social infiltration*.

Given an OSN consisting of n user accounts, Íntegro takes $O(n \log n)$ time to complete its computation. For attackers who randomly establish a set E_a of edges between victim and fake accounts, Íntegro guarantees that at most $O(\text{vol}(E_a) \log n)$ fakes are assigned ranks similar to or higher than real accounts in the worst case, where $\text{vol}(E_a)$ is the sum of weights on edges in E_a. In fact, this bound on ranking quality represents an improvement factor of $O(|E_a|/\text{vol}(E_a))$ over SybilRank. In addition, even with a uniformly random victim classifier that labels each account as a victim with 0.5 probability, Íntegro ensures that $\text{vol}(E_a)$ is at most equals to $|E_a|$, resulting in the same asymptotic bound offered by SybilRank [15].

Main Results

We evaluated Íntegro against SybilRank using real-world datasets and a large-scale deployment at Tuenti—the largest OSN in Spain with about 15 million users. We picked SybilRank because it was shown to outperform known contenders [15], including EigenTrust [60], SybilGuard [138], SybilLimit [139], SybilInfer [19], Mislove's method [122], and GateKeeper [118]. In addition, as SybilRank relies

on a user ranking scheme that is similar to ours albeit on an unweighted version of the graph, evaluating against SybilRank allowed us to clearly show the impact of leveraging victim classification on fake account detection.

Our evaluation results show that Íntegro consistently outperforms SybilRank in ranking quality, especially as the fakes infiltrate an increasing number of victims, that is, as E_a grows large. In particular, Íntegro resulted in up to 30% improvement over SybilRank in its ranking *area under ROC curve* (AUC), which represents the probability that a random real account is ranked higher than a random fake account [122]. In fact, Íntegro achieved an AUC greater than 0.92 as $|E_a|$ increased, while SybilRank resulted in an AUC as low as 0.71 under the same setting.

In practice, the deployment of Íntegro at Tuenti resulted in up to an order of magnitude higher *precision* in fake account detection, where ideally fakes should be located at the bottom of the ranked list. In particular, for the bottom 20K low-ranking users, Íntegro achieved 95% precision, as compared to 43% by SybilRank, or 5% by Tuenti's current user-based abuse reporting system. More importantly, the precision significantly decreased as we inspected higher ranks in the list, which means Íntegro consistently placed most of the fakes at the bottom of the list, unlike SybilRank. The only requirement for Íntegro to outperform SybilRank is to train a victim classifier that is better than random. This requirement can be easily satisfied during the cross-validation phase by deploying a victim classifier with an AUC greater than 0.5. In our deployment, the victim classifier was 52% better than random with an AUC of 0.76, although it was trained using low-cost features.

We implemented Íntegro on top of Mahout[8] and Giraph,[9] which are widely deployed, open-source distributed machine learning and graph processing systems, respectively. Using a synthetic benchmark of five OSNs consisting of up to 160M users, Íntegro scaled nearly linearly with number of users in the graph. In particular, for the largest graph with 160M nodes, it took Íntegro less than 30 minutes to finish its computation on a cluster of 33 commodity machines.

[8] http://mahout.apache.org

[9] http://giraph.apache.org

Contributions

In summary, this part of the dissertation makes the following contributions:

Contribution 3: *Leveraging victim classification for fake account detection.*

We designed and analyzed Íntegro—a fake account detection system that relies on a novel technique for integrating user-level activities into graph-level structures. Íntegro uses supervised machine learning with features extracted from user-level activities in order to identify potential victims in the OSN. By weighting the graph such that edges incident to potential victims have lower weights than other accounts, Íntegro guarantees that most real accounts are ranked higher than fakes. These ranks are derived from the landing probability of a modified random walk that starts from a known real account. To our knowledge, Íntegro is the first detection system that is robust against adverse manipulation of the graph, where fakes follow an adversarial strategy to befriend a large number of accounts, real or fake, in order to evade detection.

Contribution 4: *Open-source implementation and evaluation framework.*

We implemented Íntegro on top of open-source distributed systems which run on commodity machines. We publicly released Íntegro as part of two projects: (1) SyPy,[10] our single-machine comparative evaluation framework for graph-based fake account detection algorithms, and (2) GrafosML,[11] a library and tools for machine learning and graph analytics. We evaluated Íntegro against SybilRank using real-world datasets and a large-scale deployment at Tuenti. Íntegro has allowed Tuenti to detect at least 10 times more fake accounts than their current user-based abuse reporting system, in which reported users are not ranked. With an average of about 16K flagged accounts a day [15], Íntegro has saved Tuenti hundreds of man hours in manual verification by robustly ranking user accounts.

[10] http://boshmaf.github.io/sypy
[11] http://grafos.ml

Chapter 2

Social Infiltration in OSNs

Online social networks (OSNs) have attracted more than a billion active user and have become an integral part of today's Web. In the wrong hands, however, OSNs can be used to harvest private user data [6], distribute malware [132], control botnets [63], perform surveillance [95], spread misinformation [113], and even influence algorithmic trading [8]. Online attackers start their abuse by running a *social infiltration* campaign using automated fake accounts, where fakes are used to connect with a large number of users in the target OSN. Such an infiltration is required because isolated fake accounts cannot directly interact with or promote content to most users in the OSN [28].

While online attackers, also called *cyber-criminals*, were generally thought to be less financially driven in the past, a number of recent studies showed that such criminals are moving towards profitable business models [14, 38, 86]. For example, spammers that promote generic pharmaceuticals and knock-off designer products generate an estimated revenue between $12–92 million each year [77], while similar criminals that trick victims into installing ineffectual software, such as fake anti-virus tools, pulling in $5–116 million [105]. The threats these cyber-criminals pose to OSNs are aggravated by the emergence of an *underground economy*—a digital network of criminals who buy and sell goods that directly enable the abuse of OSNs for a small price. These goods include fake accounts [88], compromised

machines [14], CAPTCHA-solving services [87], and IP address proxies [53].

In this chapter, we consider profit-driven attackers who infiltrate OSNs using malicious *socialbots*—automation software that control fake accounts and perform social activities similar to those of legitimate users. In particular, we aim to address the following RQs, as introduced in Section 1.3.1:

- **RQ1:** How vulnerable are OSNs to social infiltration?
- **RQ2:** What are the security and privacy implications of social infiltration?
- **RQ3:** What is the economic rationale behind social infiltration at scale?

2.1 Background and Related Work

We next present required background and contrast related work to ours, focusing on socialbots, social infiltration, scalability of online attacks, and the underground market for OSN abuse.

2.1.1 Online Social Networks

An *online social network* (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 but make the distinction when deemed necessary. A social relationship between two users can be either bilateral such as friendships in Facebook, or unilateral such as followerships 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 (i.e., relationships) among the users. For every user $v_i \in V$, the *neighborhood* of v_i is the set that contains all users in V with whom v_i has social connections. For the users in the neighborhood of v_i, the union of their neighborhoods is the *extended neighborhood* of v_i. In Facebook,

for example, the neighborhood and the extended neighborhood of a user represent the "friends" and the "friends of friends" of that user, respectively.

2.1.2 Sybil Attacks and Social Infiltration

The *Sybil attack* [22] represents the situation where an attacker controls multiple identities, each called a *Sybil*, and joins a targeted system under these identities in order to subvert a particular service. Accordingly, we define *social infiltration* in OSNs as an instance of the Sybil attack, where an attacker employs an automation software, which is scalable enough to control many attacker-owned fake accounts, in order to connect with a large number of legitimate users in the target OSN.

Recent research indicates that social infiltration in OSNs is possible and represents an emerging threat [57, 91, 96]. For example, Bilge et al. [6] where among the first to demonstrate that users in OSNs are not cautious when accepting friend requests. In particular, the authors performed an experiment to evaluate how willing users are to accept friend requests sent by *forged* user accounts of people who were already in their friends list as confirmed contacts. They also compared that with users' response to friend requests sent by people who they do not know, that is, fake accounts posing as strangers. In their experiment, they found that the acceptance rate for forged accounts was over 60%, and about 20% for the fakes. Unlike their targeted attack, we do not expect the attacker to steal identities and forge user accounts, as this makes the attack non-scalable and more susceptible to detection [104]. Moreover, we aim to characterize 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 threat model of an attacker who is capable of social infiltration on a large scale.

2.1.3 Social Engineering, Automation, and Socialbots

The concept of *social engineering* is usually defined as the art of gaining unauthorized access to secure objects by exploiting human psychology, rather than using hacking techniques [4]. Social engineering has become more technical and com-

plex; social engineering attacks are being computerized and fully automated, and have become adaptive and context-aware [6, 13, 57, 58, 95].

Huber et al. [54] presented one of the first frameworks for automated social engineering in OSNs, where a new breed of bots can be used to automate traditional social engineering attacks for many adversarial objectives. Under this framework, a *malicious socialbot represents an automated social engineering tool* that allows an attacker to infiltrate online communities like a virtual con man, build up trust over time, and then exploit it to elicit information, sway opinions, and call to action. In fact, automation has a strong economic rationale behind it. Herley [49] showed *for an online attack to be scalable, it ought to be automated without manual per-user adjustments*. If not, there are no economic incentives for a *rational* (i.e., profit-driven) attacker to scale the attack, which is typically undesirable from an adversarial standpoint.

Socialbots can be used for non-adversarial objectives as well [56]. For example, the Web Ecology Project[1] envisions the design of benign socialbots that have positive impact on online communities by advocating awareness and cooperation among human users on civic or humanitarian issues. Soon after, this objective was extended towards realizing *social architecture* [92], where "intelligent" socialbots are used to interact with, promote, and provoke online communities towards desirable behaviors, including large-scale restructuring of social graphs.

2.1.4 Online Attacks in a Web of Scale

The distinction between online attacks that are scalable and those that are not has long been recognized. Dwork and Naor [23] suggested that forcing a linear cost dependence makes certain attacks financially unattractive.

Similarly, Herley [49] distinguished between *scalable* attacks, in which costs are almost independent of the number of attacked users, and *non-scalable* or *targeted* attacks, which involve per-user effort resulting in a higher than average cost. To compensate, the non-scalable attacker must target users with higher than aver-

[1] http://www.webecologyproject.org

age value, as low value users negatively affect the reward. To accomplish this, the non-scalable attacker needs that value be both visible and very concentrated, with few users having very high value while most have little. In this the attacker is fortunate; power-law long-tail distributions that describe the distributions of wealth, fame, and other phenomena are extremely concentrated. However, in these distributions, only a small fraction of the population have above average value. For example, fewer than 2% of people have above average wealth in the US [20]. Thus, when attacking assets where value is concentrated, the non-scalable attacker ignores the vast majority of users. By contrast, the scalable attacker lives in a "costs nothing to try" world and attacks everyone indiscriminately, whether they have high or low value, ending up with an average value per user. As a result, *scalable attacks reach orders of magnitude more users than non-scalable ones*. This, to some extent, explains why only few users ever get targeted with sophisticated attacks, while mostly all users receive spam emails at some point in time [49].

The scale of an attack, interestingly, can lead to its own demise. Florêncio et al. [35] observed that there is an enormous gap between potential and actual harm of scalable online attacks; the majority of Web users appear unharmed each year. The authors explained that *a scalable attacker faces a sum-of-effort rather than a weakest-link defense*, leading to an increased cost per user. As scalable attacks must be profitable in expectation, not merely in particular scenarios, many attacks can be rendered non-profitable and non-scalable by marginally increasing the cost to subvert the defenses deployed by the service provider or its users, even when many profitable targets exist.

We analyze social infiltration in OSNs from an economic perspective, focusing on the *scale* of infiltration, in terms of number of attacked users, and its effect on profitability. We extend Herley's analysis and assume the role of an attacker who derives profit-maximizing infiltration strategies under different scalability requirements. As we discuss next, social infiltration and other online attacks, scalable or not, integrate into a cyber-criminal ecosystem for OSN abuse, with a thriving underground market pulling in hundreds of millions of dollars in revenue.

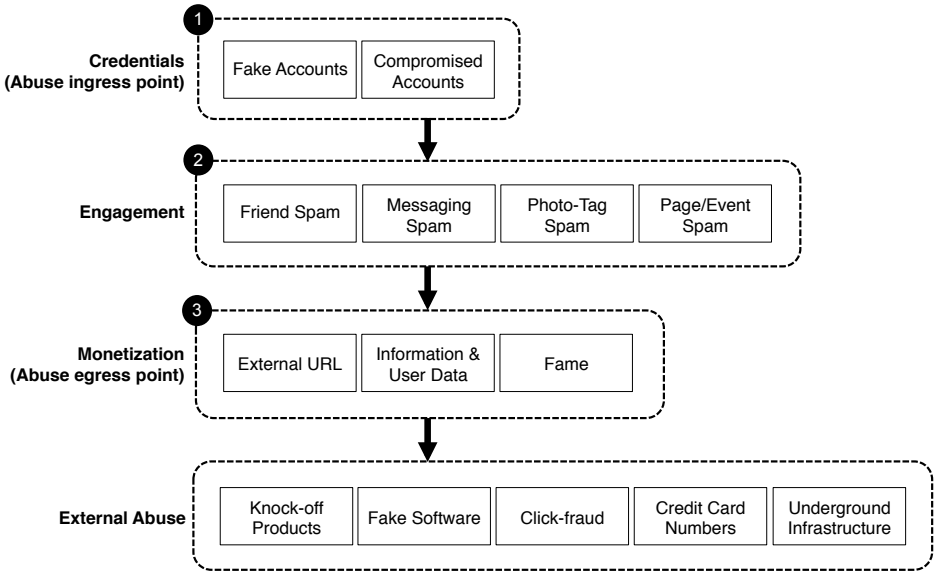


Figure 2.1: Components required for OSN abuse in Facebook

2.1.5 The Cyber-Criminal Ecosystem for OSN Abuse

As OSNs dominate the daily activities of Web users, cyber-criminals have adapted their monetization strategies to engage users within these walled gardens. Attacks targeting OSNs require three components [114]: (1) access to account credentials, (2) a mechanism to engage with legitimate users within the network (i.e. the victims that will be exploited to realize a profit), and (3) some form of a monetizable content. With respect to Facebook, Figure 2.1 shows the underpinnings of each component, which we use to guide the upcoming discussion.

As the figure shows, an attacker can use fake or compromised accounts in order to get access to Facebook. Shortly after that, to draw an audience, the attacker can engage users by running a social infiltration campaign (i.e., by sending friend request spam). If user privacy settings are set to "public to everyone," which is not usually the case [46, 68], the attacker can engage users through other kinds of public communication, such as messaging, photo-tagging, and page/event requests. In order to monetize a victim, users are typically directed away from Facebook via a URL to another website, which generates a profit via knock-off products, fake

software, click-fraud, banking information, or a malware that converts a victim's machine or assets (e.g., credentials, private data) into a commodity for the underground market (e.g., zombie machines, compromised accounts, email addresses).

At the heart of this for-profit cyber-criminal ecosystem is an underground market that connects attackers with parties selling a range of specialized products and services, including spam hosting [3], fake accounts [88, 114], compromised zombie machines [14], CAPTCHA-solving services [87], IP address proxies [53], and exploit kits [42]. Even simple services such as generating favorable reviews or writing webpage content are for sale [126]. Revenue generated by criminals participating in this market varies widely based on business strategy, with spam affiliate programs generating $12–92 million [77] and fake anti-virus scammers pulling in $5–116 million [105] over the course of their operations.

Within this cyber-criminal ecosystem for OSN abuse, the analysis we present in Section 2.5 suggests that social infiltration at scale is not financially attractive as an independent business. However, it facilitates a *non-zero profit monetization campaign* for underground market commodities, as illustrated in Figure 2.1.

2.2 Vulnerabilities in OSN Platforms

We discuss four vulnerabilities found in today's OSNs that allow an attacker to run a large-scale social infiltration campaign. Collectively, along with poorly designed end-user privacy controls [73], these vulnerabilities represent the *enabling factors* that make operating socialbots feasible in the first place.

2.2.1 Ineffective CAPTCHAs

Typically, OSNs employ CAPTCHAs [124], which are a type of challenge-response test used in computing to determine whether or not the user is human, in order to prevent automated bots from abusing their platforms. Attackers, however, can often circumvent this countermeasure using different techniques, such as automated analysis using optical character recognition and machine learning [6], exploiting

botnets to trick victims into manually solving CAPTCHAs [25], reusing session identifiers of known CAPTCHAs [52], or even hiring cheap human labor [87].

Let us consider the use of human labor to solve CAPTCHAs; a phenomenon known as a CAPTCHA-solving business. Motoyama et al. [87] showed that companies involved in such a business are surprisingly efficient: (1) they have high service quality with a success rate of up to 98% in solving CAPTCHAs, (2) they charge less than $1 per 1K successfully solved CAPTCHAs, and (3) they provide software APIs to automate the whole process. Thus, even the most sophisticated CAPTCHA that only humans could solve can be effectively circumvented with a small investment from an attacker. In such a situation, the attacker is considered a rational agent who invests in such businesses if the expected return on investment is considerably high. This also *allows researchers to study online attacks from an economic context*, and define cost structures that measure when it is economically feasible for an attacker to mount scalable attacks that involve, for instance, solving CAPTCHAs by employing cheap human labor [49]. We provide such an analysis for social infiltration at scale in Section 2.5.

2.2.2 Fake Accounts and User Profiles

Creating a new user account on an OSN involves three tasks: (1) providing an active email address, (2) creating a user profile, and (3) solving a CAPTCHA if required. Each user account maps to one profile, but many user accounts can be owned by the same person or organization using different email addresses, which represents a potential Sybil attack. In what follows, we discuss how an attacker can fully automate the account creation process in order to create a group of fake accounts, where each account is represented by a fake user profile. In fact, this automation is not new as similar tools, such as FriendBomber,[2] are used for online marketing. The attacker can write a customized software to create fake accounts or buy OSN accounts in bulk from underground markets [88].

[2] http://www.friendbomber.com

Fake Accounts

When creating a new user account, an email address is required to validate then activate the account. Usually, the OSN validates the account by associating it with the owner of the email address. After account validation, the owner activates the account by following an activation link that is emailed by the OSN. Accordingly, an attacker has to overcome two hurdles when creating a new fake account: (1) providing a working email address which is under his control, and (2) performing email-based account activation. To tackle the first hurdle, the attacker 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 for the second hurdle, an attacker 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.

Fake User Profiles

Creating a user profile is a straightforward task for legitimate users, as they just have to provide the information that describes their *social attributes* (e.g., name, age, gender, interests). For an attacker, however, the situation is subtly different. The objective of the attacker is to create profiles that are "socially attractive." We consider a purely adversarial standpoint concerning social attractiveness, where attackers aim 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]. Thus, an attacker can use publicly available personal pictures for the newly created user profiles, along with the corresponding gender and age range information. The attacker can use already-rated personal

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

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 attacker to automate the collection of required profile information in order to populate a fake user profile by crawling, or scavenging, the Web.

2.2.3 Crawlable Social Graphs

The social graph of an OSN is usually hidden from public access in order to protect its users' privacy. An attacker, however, can reconstruct parts or a complete version of the social graph by first logging in to the OSN platform using one or many fake 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 attacker to parse the connections lists of user profiles, such as the "friends list" in Facebook, along with their profile information. After that, the attacker can gradually construct the corresponding social graph with accessible social attributes using an online search algorithm, such as breadth-first search [80]. The attacker can build either a customized web crawler for this task or resort to cheap commercial crawling services, such as 80Legs,[6] that support social-content crawling.

2.2.4 Exploitable Platforms and APIs

Most OSNs provide software APIs that enable the integration of their platforms into third-party software systems. Facebook's Graph API [67], for example, enables third parties to read from and write data into Facebook's platform. It also provides a simple and consistent view of Facebook's social graph by uniformly representing objects (e.g., profiles, photos, posts) and the connections between them (e.g., friendships, likes, tags). An attacker, however, can use such APIs to automate the execution of social activities online. If an activity is not supported

[5] http://hotornot.com
[6] http://80legs.com

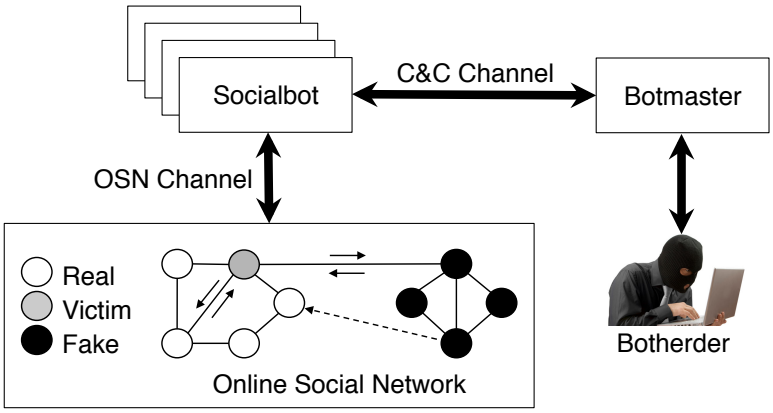


Figure 2.2: An overview of a socialbot network (SbN)

by the API, the attacker can scrape the content from the platform's website, and record the exact HTTP requests which are used to carry out such an activity in order to create 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 attacker, however, can always choose to reduce the frequency at which the bots sends the requests to avoid CAPTCHAs. Another technique is to inject artificial connection requests into non-encrypted OSN traffic at the HTTP level, so that it would appear as if users have added the socialbot as a contact [55].

2.3 Design of a Social Botnet

In what follows, we present the design of a web-based social botnet. In particular, we start with an overview of the socialbot network (SbN) concept and define its threat model. This is followed by a detailed discussion of the SbN design and its main components.

2.3.1 Overview

We studied large-scale infiltration in OSNs as an organized campaign run by a network of malicious socialbots. We followed the design of web-based botnets [101] and defined a *socialbot network* (SbN): A group of malicious socialbots that are orchestrated by an attacker called the *botherder*.

As shown in Figure 2.2, an SbN consists of a set of socialbots, a botmaster, and a command & control (C&C) channel. Each socialbot controls a fake account in a target 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 friend request). These commands are either sent by the botmaster or defined locally on each socialbot. All data collected by socialbots are called the *botcargo*, and are always sent back to the botmaster. A *botmaster* is a software controller with which the botherder interacts in order to define commands that are sent through the C&C channel. We designed the botmaster to exploit known user behaviors in OSNs, such as the *triadic closure principle* [24], in order to improve the success rate of the infiltration.[7] The *C&C channel* is a communication medium that facilitates the transfer of the botcargo and the commands between the socialbots and the botmaster, including any *heartbeat signals*, which are used to check whether a socialbot is online and operational.

In Figure 2.2, each node in the OSN represents a user account. Fake accounts, which are controlled by socialbots, are marked in black. Victim accounts, which are real accounts controlled by legitimate users but have been infiltrated by fakes, are marked in grey. Edges between nodes represent bilateral social connections. The dashed arrow represents a connection request. Small arrows represent social interactions, such as posting a message. As we explain next, the SbN can be part of an existing botnet, where compromised machines are also infected by the socialbot malware that control fake accounts in the target OSN.

[7] The triadic closure principle states that if two people have a friend in common, then there is an increased likelihood that they will become friends themselves in the future.

2.3.2 Threat Model

We assume a global passive attacker who is capable of designing and operating a fully or semi automated SbN on a large scale. This involves exploiting all of the vulnerabilities presented in Section 2.2, which collectively enable the operation of an SbN in the target OSN. We also assume the attacker is capable of deploying the SbN as part of an existing botnet, and thus, we treat the SbN as a distributed network of compromised "zombie" machines acting cooperatively. Accordingly, we believe it is fair to assume that the defenders, consisting of OSNs and Internet service providers, are able to cooperate, and therefore have a global view of the communication traffic. Finally, we 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 2.4.

As for the adversarial objectives, the botherder designs and operates an SbN to (1) carry out a social infiltration campaign in the target OSN, and to (2) harvest private user data. The first objective involves connecting with a large number of either random or targeted OSN users for the purpose of establishing an influential position, which can be exploited to promote malicious content or spread misinformation. In the second objective, however, the botherder aims to generate profit by collecting personally identifiable information that have monetary value as goods in underground markets. In addition, these data can be used to craft personalized messages for subsequent spam, phishing, or astroturfing campaigns.

2.3.3 Requirements

Ideally, an SbN has to be automated and highly scalable to control hundreds of socialbots, which is achieved by following the design of web-based botnets. In order to be effective, however, an SbN has to meet three challenging requirements: (1) each socialbot has to be designed in such a way that hides its true face, that is, to pass itself off as a human not a bot, (2) the botmaster has to implement strategies

Operation	Type	Description
`read(o,p)`	Interaction	Reads object `o` from account `p` and returns its value `v` as botcargo
`write(v,o,p)`	Interaction	Writes value `v` to object `o` on account `p`
`connect(b,p)`	Structure	Sends or accepts a connection request between accounts `b` and `p`
`break(b,p)`	Structure	Breaks the social connection between accounts `b` and `p`

Table 2.1: Generic operations supported by the socialbot malware

that enable large-scale social infiltration in the target OSN, and (3) the traffic in the C&C channel has to look benign in order to avoid detecting the botmaster.

We decided to use a simplistic design in order to meet each one of these requirements. As we describe next, we used techniques that have shown to be both feasible and effective. We acknowledge, however, that more sophisticated techniques, which use machine learning algorithms [74], are possible. We refrained from using such techniques as our goal is to evaluate the threat of social infiltration and characterize user behaviors, rather than to optimize the performance of an SbN. We discuss the details of the used techniques in what follows.

2.3.4 Socialbots

A socialbot consists of two components: (1) a fake user account on a target OSN, and (2) the socialbot software. As we design the socialbot software in an adversarial setting, we regard this software as being malicious and refer to it as a *malware*. We next present the primitives required for the socialbot malware to mimic real user behavior in OSNs (e.g., posting messages, befriending others).

Each socialbot malware supports two types of *generic operations* in any given OSN: (1) social interaction operations that are used to read and write social content, and (2) social structure operations that are used to modify the social graph. A detailed description of these operations is shown in Table 2.1.

We define a set of *commands* that each includes a sequence of generic operations. A command is used to mimic a real user action that relates to social content generation and networking, such as posting a status update and joining a commu-

nity of users. A command is either native or master, depending on its origin. A *native* command is defined locally on each socialbots, while a *master* command is sent by the botmaster to the socialbot through the C&C channel. For example, we define a single native command called `update(b)`, which is executed by each socialbot, as follows: At arbitrary times, each socialbot `b` generates a message `m` and executes the operation `write(m,o,b)`, where `o` is the object which maintains messages for the account `b`. This command resembles a user posting a status update message on her profile, and is executed at arbitrary times in order to avoid creating detectable patterns. More sophisticated commands can be defined that, for example, allow socialbots to comment on each others' status updates.

Each socialbot employs a native *controller*: A two-state finite-state machine (FSM) that enables the socialbot to either socialize with others by executing commands or stay dormant. State transition occurs whenever a native or master command needs to be executed. A native controller can also enable advanced social interaction capabilities by integrating existing chatter bots [54] or hijacking online human-to-human conversations in a man-in-the-middle fashion [69].

2.3.5 Botmaster

A botmaster is an automation software that orchestrates the overall operation of an SbN. The botmaster software consists of a botworker, a botupdater, and a C&C engine. The *botworker* maintains socialbots in the SbN by creating fake accounts and delegating each account's credentials, the user name and password, to the socialbot's malware in order to get full control over the account. The *botupdater* pushes software updates to socialbots using the C&C channel, including new native commands, modified HTTP-request templates, and improved CAPTCHA-solving capabilities. The *C&C engine* manages a repository of master commands and deploys 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. We next present a set of master commands that

Command	Description
`cluster(b,k)`	Connects socialbot `b` with at most `k` other socialbots
`seed(b,k)`	Connects socialbot `b` with `k` user profiles that are picked at random
`decluster(b)`	Breaks the social connections between socialbot `b` and other socialbots
`collect(b)`	Returns profile information of users in the neighborhoods of socialbot `b`
`exploit(b)`	Connects socialbot `b` with users in its extended neighborhood

Table 2.2: Master commands executed by socialbots

define a social infiltration strategy which exploits known user behaviors in OSNs. These master commands, which are summarized in Table 2.2, are issued by the master controller in three super-states, or *phases*, as follows:

Setting up the SbN

In the beginning, each socialbot has no connections and is isolated from the rest of the users in the targeted OSN, which is not favorable for social infiltration. In particular, Tong et al. [116] showed 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 OSN's average. Therefore, in an attempt to increase the social attractiveness of socialbots, we define a master command `cluster(b,k)`, which orders the socialbot `b` to connect with at most `k` other socialbots. This initial clustering of socialbots can be helpful in evading OSN security defenses that keep track of how many rejected connection requests each new user has after joining the OSN, which is usually used as an indication of an automated activity by fake accounts [135].

Seeding the Infiltration

After initial setup, the socialbots start their social infiltration campaign. In order to bootstrap the infiltration with some victims, the *seeds*, we define a master command `seed(b,k)`, which orders each socialbot `b` to connect with `k` user profiles that are picked at random from the targeted OSN. Because the clique structure among the socialbots can be easily detected [137], we define a master command

`decluster(b)` that orders the socialbot `b` to break the social connections with all other socialbots. One can define a similar master command which orders each socialbot to break one social connection with another socialbot for every new victim, and thus gradually decluster.

To satisfy the second objective in the threat model, we define the master command `collect(b)`, which orders the socialbot `b` to collect accessible user profile information in the direct and extended neighborhoods of its victims, and then return them as botcargo. This command is issued whenever a new user is infiltrated.

Exploiting Triadic Closure

It has been widely observed that if two users have a friend in common, then there is an increased chance that they become friends themselves in the future [71]. This property, which is also known as the *triadic closure principle* [24], was first observed in real-world social networks. Nagle et al. [91] showed that the likelihood of accepting a friend request in Facebook is three times higher given the existence of some number of mutual friend. Accordingly, in order to increase the yield of social infiltration in the target OSN, we define a master command `exploit(b)`, which orders the socialbot `b` to connect with users with whom it has some mutual connections, that is, users in its extended neighborhood.

2.3.6 C&C Channel

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

As for the C&C channel, we recall that detecting the botmaster from the C&C traffic is as hard as it is in a traditional botnet, as the botherder can rely on an existing botnet infrastructure and deploy the SbN as part of the botnet, as discussed

in Section 2.3.2. Alternatively, the botherder can exploit the OSN platform itself for the C&C infrastructure [63]. For example, Nagaraja et al. [90] showed that a botherder can establish a probabilistically unobservable C&C channel by building a covert OSN botnet that, for example, uses image steganography to hide the communication as part of photo sharing behavior of OSN users.

2.4 Empirical Evaluation

To evaluate how vulnerable OSNs are to social infiltration, we implemented an SbN according to the discussion in Section 2.3. We picked Facebook as the target OSN because it is the largest OSN today, consisting of more than one billion active user [30]. Moreover, unlike other OSNs, Facebook is mostly used to connect with real-world friends and family not with strangers [27, 68]. Finally, Facebook employs a state-of-the-art "immune system" that detects thousands of fake accounts a day using various machine learning techniques [104]. Therefore, the success of social infiltration on an OSN such as Facebook represents a serious threat which needs to be addressed.

2.4.1 Ethics Consideration

Given the nature of social infiltration, one legitimate concern is whether it is ethically acceptable and justifiable to conduct such a research experiment. As computer security researchers, we believe that controlled, minimal-risk, and realistic experiments are the only way to reliably estimate the feasibility of cyber attacks in the real-world. These experiments allow us and the wider research community to gain a genuine insight into the ecosystem of online attacks, which is useful in understanding how similar attacks may behave and how to defend against them.

We carefully designed our experiment in order to reduce any potential risk at the user side [9]. In particular, we followed known practices and received the approval of our university's behavioral research ethics board (BREB).[8] In addition,

[8]This study has been approved by BREB application #H10-01439.

we strongly encrypted and secured all collected data.

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 reported these vulnerabilities to Facebook through its platform's vulnerability reporting tool [31].

2.4.2 Methodology

Our main research objective is to characterize users' response to a social infiltration campaign in OSNs, along with the corresponding security and privacy implications. We implemented an SbN prototype targeting Facebook for the reasons outlined above, and operated this SbN for 8 weeks during the first quarter of 2011. The duration of the experiment was informed by how many data points we needed to properly capture user 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 Internet service provider. We also implemented a simple HTTP proxy on the used machine in order to make the traffic appear as if it originated from multiple clients having different browsers and operating systems. Even though the university-assigned IP address might have diluted Facebook's immune system, we believe that it is unsafe to completely white-list university IP addresses.[9] 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 [101].

[9]While operating the SbN under the university-issued IP address, we observed that some operations were identified as malicious, and the used IP address was temporarily blocked, especially during fake account creation. This supports the argument that even university IP addresses were audited by Facebook, and they were not fully white-listed.

2.4.3 Implementation on Facebook

In our prototype's implementation, each socialbot ran the same malware and was equipped with one native command, namely, `update`. We implemented the generic operations described in Table 2.1 using API calls and HTTP-request templates, as follows. First, we used Facebook's Graph API [67] to carry out social interaction operations. The API, however, requires the user—the socialbot in this case—to be logged in to Facebook at the time of any API call. As login sessions had timeouts, we instead developed a Facebook application that fetched permanent OAuth 2.0 access tokens in order to allow each socialbot to send API calls without the need to login.[10] Second, for social structure operations, we used prerecorded HTTP-request templates that enabled each socialbot to send friend requests to other users, as if these requests were sent from a browser. To generate status update messages, we used an API provided by iHeartQuotes[11] to pull random quotes and blurbs.

For the botmaster software, our implementation integrates the botworker with the following websites: DeCaptcher,[12] a CAPTCHA-solving business; HotOrNot, a photo-sharing website; and MailRu, a web email provider. The implementation also supports on-demand updates through the botupdater, including HTTP-request templates and native commands. Finally, we implemented each master command described in Table 2.2.

Let us consider the random sampling used in `seed(b,k)`. On Facebook, each user profile has a unique identifier (ID) represented by a 64-bit integer, which is assigned at account creation time. In order to obtain a uniformly random sample of Facebook users, we used a sampling technique called *rejection sampling* [99], as follows. First, we randomly generated 64-bit integers from known ID ranges used by Facebook [39]. After that, we checked whether each newly generated ID mapped to an existing user profile by probing the corresponding profile page. If the profile existed, we included the user in the random sample only if the profile

[10] The application was called "Desktop Connector," and was hosted on the same machine.
[11] http://iheartquotes.com
[12] http://de-captcher.com

was not isolated. We define an *isolated* user profile as a profile that does not have friends or does not share its friends list on Facebook.

We implemented the simple native and master controllers that follow a 3-phase infiltration strategy. We note, however, that a more resourceful attacker can employ *adversarial classifier reverse engineering* [74] techniques to derive an infiltration strategy that has a higher chance at evading automated detection by defense systems similar to those deployed by Facebook.

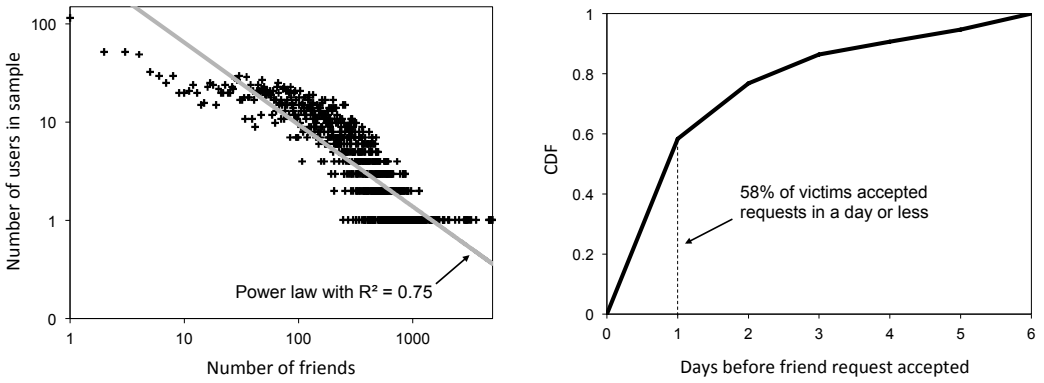
2.4.4 Experimentation

We operated the SbN prototype for 8 weeks from 28 January to 23 March, 2011. The socialbots send 9,646 friend requests out of which 3,439 requests were accepted by legitimate users, resulting in an *average acceptance rate* of 35.7% . We refer to such infiltrated users who accepted friend requests sent by malicious socialbots as *victims*. We divide the upcoming discussion according to the 3-phase social infiltration strategy.

Phase 1 – Setup

Our SbN prototype, which was physically hosted on a single commodity machine, consisted of 100 socialbots and a single botmaster.[13] Even though we could have automatically created fake accounts for the socialbots, we decided not to financially support the CAPTCHA-solving business. In total, we manually created 49 socialbots that had male user profiles and 51 socialbots that had female user profiles. The socialbots clustered into a 100-clique structure, representing a tightly-knit, cohesive community of users, as discussed in Section 2.3.5

[13] We note that the original study [10] involved two more socialbots. As these socialbots did not participate in the infiltration but were rather used to check the status of the SbN, we decided to exclude them from our analysis herein.



(a) Degree distribution of user sample **(b)** Victims response time to requests

Figure 2.3: Seeding social infiltration in Facebook

Phase 2 – Seeding

The socialbots generated a random sample of 5,053 valid user profile IDs. These unique IDs passed the inclusion criteria we presented in Section 2.4.3. Figure 2.3a shows the *degree distribution* of this uniform sample.[14]

Each socialbot sent 25 friend requests per day in order to avoid CAPTCHAs. Accordingly, it took the bots 2 days to send friend requests to the sampled users, where each user received exactly one request. In total, 2,391 requests were sent from "male" socialbots and 2,662 from "female" socialbots. We kept monitoring the status of each request for 6 days after being sent. Overall, 976 requests were accepted, resulting in an average acceptance rate of 19.3%. In particular, out of all accepted requests, 381 were sent from male socialbots (15.9%) and 595 were sent from female socialbots (22.3%). The difference in the average acceptance rate was statistically significant ($\chi^2 = 32.8702$, $p = 9.852 \times 10^{-9}$, CI=95%), where female socialbots outperformed male socialbots by 6.4%, on average.[15] Moreover, 58% of the victims accepted the request during the same day of being sent, as shown in Figure 2.3b. In our implementation, the socialbots gradually broke the 100-clique

[14]The degree of a node is the size of its neighborhood. The degree distribution is the probability distribution of these degrees over all nodes included in the sample.

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

structure as they infiltrated more victims. The SbN spent 2 weeks in this seeding phase. For most of that time, however, the SbN was setting idle.

Phase III – Exploitation

We kept the SbN running for another 6 weeks on Facebook. During this time, the socialbots send another $4,593$ friend requests to users in their extended neighborhoods. Overall, $2,463$ requests were accepted by victims, resulting in an average acceptance rate of 53.6%. The difference in average acceptance rate between the two phases was statistically significant ($\chi^2 = 1429.9$, $p = 2.2 \times 10^{-16}$, CI=95%), where exploiting triadic closure resulted in 34.3% higher average acceptance rate than targeting users at random.

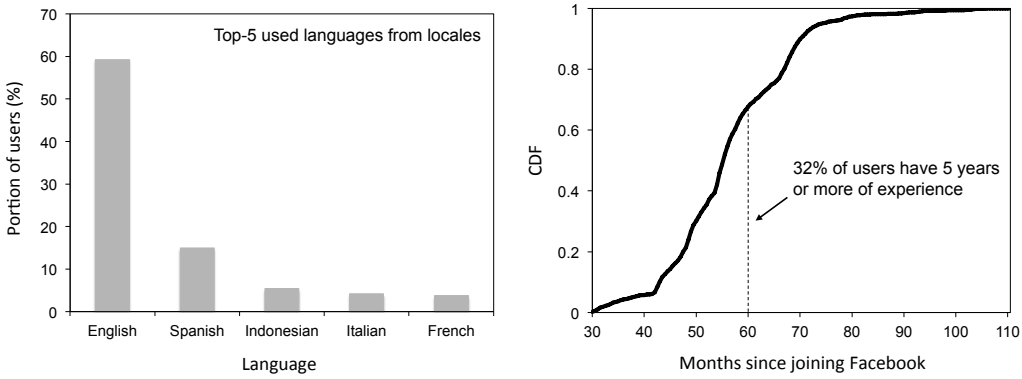
2.4.5 Analysis and Discussion

In what follows, we analyze and discuss the results presented in the Section 2.4.4. We focus on characterizing user behaviors, the infiltration performance, and its implications on user privacy and fake account detection mechanisms.

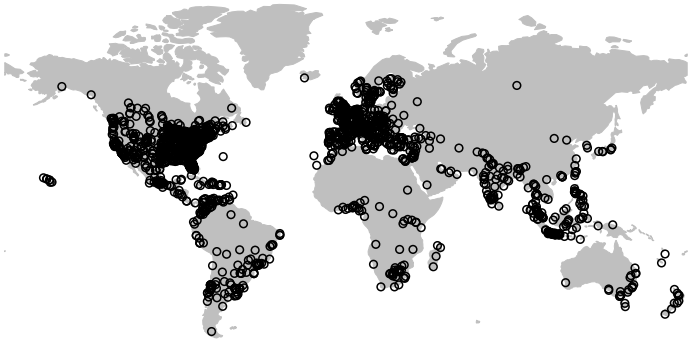
User Behavior

As the infiltration was seeded at random, the 9,646 users who received friend requests sent by socialbots represented a diverse sample of Facebook's user population. In particular, the users were 51.3% males and 48.7% females, lived in 1,983 cities across 127 countries, practiced 43 languages, and have used Facebook for 5.4 years on average, as shown in Figure 2.4.

The main observation of this study is that *some users are more likely to become victims than others*. First, in the seeding phase, we found that the more friends a user had, the more likely the user was to accept friend requests sent by socialbots posing as strangers, regardless to their gender or mutual friends, as shown in Figure 2.5a. Second, in the exploitation phase, we found that the more mutual friends the user had with a socialbot, the more likely the user was to accept friend request send by this socialbot, as shown in Figure 2.5b.

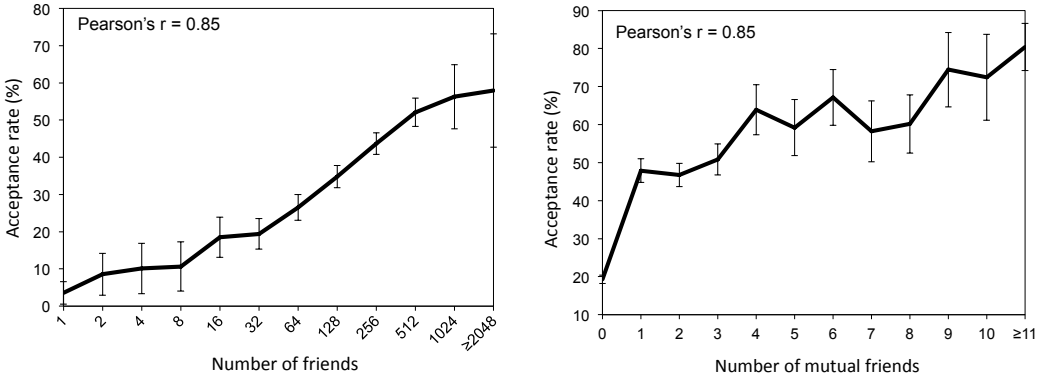


(a) Language **(b)** Experience



(c) Location

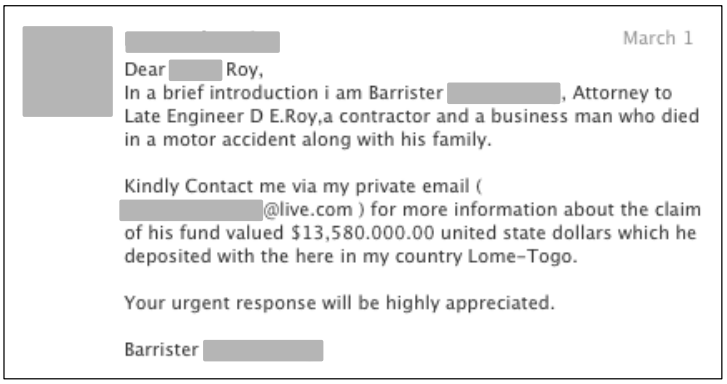
Figure 2.4: Demographics of contacted users



(a) Seeding at random **(b)** Exploiting triadic closure

Figure 2.5: User susceptibility to infiltration on Facebook (CI=95%)

(a) Victim's comment on a status update posted by a socialbot



(b) Personalized phishing message sent to a socialbot

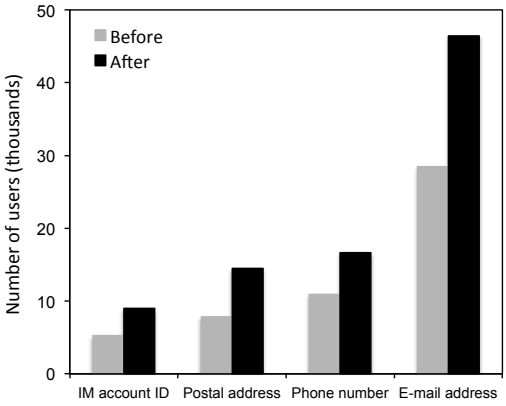
Figure 2.6: Real-world samples of user interactions with socialbots

Most of the victims maintained a social history with the socialbots. In particular, the socialbots received 73 personal messages, 112 "likes," and 164 "wall" posts by their victims. The bots also received 331 friend requests from the friends of their victims, that is, from users in their extended neighborhoods. Interestingly, the socialbots received 2 messages and 8 wall posts from users who are not in their friends list. Based on manual inspection, we found that all of them were in fact malicious, and were sent by a fake or hijacked user account.[16] Figure 2.6 shows a sample of user interactions with socialbots.

[16]This was possible because the accounts controlled by socialbots had public user profiles.

Profile Info	Direct (%) Before	Direct (%) After	Extended (%) Before	Extended (%) After
Birth Date	3.5	72.4	4.5	53.8
Email Address	2.4	71.8	2.6	4.1
Gender	69.1	69.2	84.2	84.2
Home City	26.5	46.2	29.2	45.2
Current City	25.4	42.9	27.8	41.6
Phone Number	0.9	21.1	1.0	1.5
School Name	10.8	19.7	12.0	20.4
Postal Address	0.9	19.0	0.7	1.3
IM Account ID	0.6	10.9	0.5	0.8
Married To	2.9	6.4	3.9	4.9
Worked At	2.8	4.0	2.8	3.2
Average	13.3	34.9	15.4	23.7

(a)



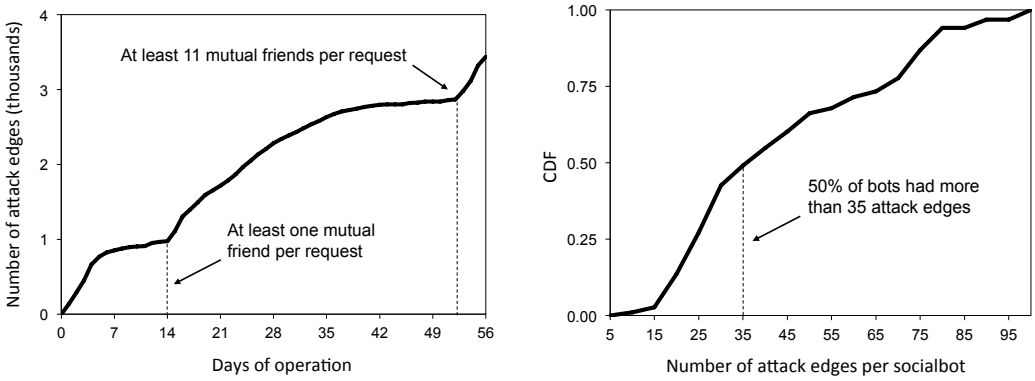
(b)

Figure 2.7: Users with accessible private data

Collected Data

The socialbots harvested large amounts of user data through the `collect` master command. By the end of the 8th week, the SbN resulted in a total of 250GB inbound and 3GB outbound network traffic between our machine and Facebook. We were able to collect news feeds, user profile information, and wall posts. In other words, practically everything shared on Facebook by the victims, which could be used for large-scale user surveillance [95]. Even though adversarial surveillance, such as online profiling [44], is a serious threat to user privacy, we decided to focus on user data that have monetary value at underground markets, such as personally identifiable information (PII).

After excluding remaining friendships among the bots, the size of their direct neighborhoods was 3,439 users. The size of all extended neighborhoods, however, was as large as 1,085,785 users. In Figure 2.7a, we compare data revelation before and after operating the SbN in terms of percentage of users with accessible private data, PII in particular. *On average, 2.62 times more PII was exposed in direct neighborhoods after infiltration, and 1.54 times more in extended neighborhoods*. While one would expect the averages of direct and extended neighborhoods to be approximately the same before the infiltration, there is a small difference of 2.1%.



(a) Infiltration over time **(b)** Infiltration per socialbot

Figure 2.8: Infiltration performance on Facebook

This minor artifact can be attributed to the relatively smaller sample size of direct neighborhoods and the non-uniform sampling method of extended neighborhoods.

Infiltration Performance

The socialbots infiltrated users over a period of 55 days, starting January 28, 2011. During this time, the bots established 3,439 friendships with victim users, where each friendship or *attack edge* connects exactly one victim to a socialbot, as shown in Figure 2.8a. The figure also illustrates the effect of triadic closure. In particular, the infiltration rapidly increased after the first 2 weeks, where each socialbot had at least one friend in common with the user to which it sent a friend request.

Another observation of this study is that *attack edges are generally easy to establish*. As shown in Figure 2.8b, an attacker can establish enough attack edges such that fake accounts, which are controlled by socialbots, are strongly connected to real accounts in the social graph. This observation has *serious implications on the performance of graph-based fake account detection systems* such as EigenTrust [60], SybilLimit [139], SybilInfer [19], GateKeeper [118], and SybilRank [15]. In particular, such systems often assume that fake accounts can establish only a small number of attack edges, ideally at most one per fake [137], so

that the *cut* which crosses over attack edges is sparse.[17] Given this assumption, these systems attempt to find such a sparse cut with formal guarantees in order to separate fakes from real accounts. To our knowledge, we are the first to provide an empirical evidence that invalidates this assumption. We systematically evaluate the implications of this finding on detection accuracy in Chapter 3.

Finally, we found that only 20 fake accounts were blocked by Facebook (20%), and curiously, all of these accounts represented female user profiles. After further investigation, we found that these accounts were blocked because concerned Facebook users flagged them as spam.[18] In fact, *we did not observe any evidence that Facebook's immune system detected the social infiltration campaign*, other than relying on user feedback. We also noticed that accumulating social history using the `cluster` and `update` commands was in fact beneficial. In particular, when we created 15 fake accounts for a pilot study in which the socialbots did not use these commands, 93.3% of fakes were blocked by Facebook in less than a week.

2.5 Economic Feasibility

In what follows, we analyze the economic feasibility of social infiltration at scale. We consider a *rational* botherder who runs a social infiltration campaign using a socialbot network (SbN) in a target OSN. Unlike a botherder who is motivated by fun, self-fulfillment, or proof of skills, a rational botherder makes decisions about the SbN operation and its infiltration strategy based on whether the expected reward exceeds the total cost. In particular, we seek to understand how the *scale of a social infiltration*—number of attacked users who received friend requests from fakes—affects its profit and infiltration strategies.

Our analysis suggests that social infiltration at scale, while not financially attractive by itself, is a crucial part of the larger cyber-criminal underground market for OSN abuse. In particular, it plays the role of a market enabler of the actual

[17] A cut is a partition of the nodes of a graph into two disjoint subsets. Visually, it is a line that cuts through or crosses over a set of edges in the graph.

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

"money-makers," which include social spam and malware, as confirmed by various empirical studies on Twitter [109, 114].

2.5.1 Methodology

We developed a simple economic model using cost-volume-profit (CVP) analysis (Section 2.5.2). We chose CVP analysis because it is typically used for short-run decisions, which suites the underground market for OSN abuse that suffers from cheating, dishonesty, and uncertainty [50]. Moreover, CVP analysis assumes the behavior of costs and rewards does not change during the infiltration campaign, which we believe is a reasonable assumption given the lack of information about volume dynamics and discounts in underground markets.

We used this economic model to define and analyze the *scale* of a social infiltration campaign, quantified by the number of attacked users, from an economic perspective. In particular, we analyzed the cost structure of an SbN as the infiltration grows arbitrarily large in scale, focusing on its effect on profitability (Section 2.5.3). Using this model, we derived profit-maximizing infiltration strategies that implement two business models under different scalability requirements (Section 2.5.4). We applied these strategies in practice and found that they lead to a non-zero profit on Facebook when operated as part of the underground market for OSN abuse (Sections 2.5.5 and 2.5.6).

2.5.2 Model and Assumptions

We assume the threat model described in Section 2.3.2. In particular, we consider a rational botherder who operates an SbN with scale n and size m. The *scale* represents the number of *attacked users*—users who receive connection requests from socialbots. The *size* represents the number of socialbots in the SbN.

The objective of the botherder is to maximize the profit by operating the SbN under the best profit-maximizing infiltration strategy. Each socialbot has a fixed average cost $\bar{c}$ associated with it, which represents the estimated cost of creating a new fake account. Moreover, each socialbot can have up to k social connections,

where k is an OSN-specific parameter (e.g., 5K in Facebook). Each user profile in the target OSN has an associated average extracted value $\bar{v}$, which represents, for example, an estimate of the monetary value of the profile's information.

We denote the *reward* and *total cost* of running a social infiltration campaign with scale n by $R(n)$ and $C(n)$, respectively. Operating an SbN is profitable only if the reward exceeds the total cost, that is, whenever the *profit* $P(n)$ is *non-zero* (i.e., positive), as follows:

$$P(n) = R(n) - C(n) > 0, \tag{2.1}$$

where the botherder *breaks even* whenever $n = n_0 > 0$ such that $P(n_0) = 0$.

2.5.3 Scalability of Social Infiltration

Informally, we say that an SbN is scalable if the botherder has an economic incentive to attack ever more users in a single infiltration campaign, that is, whenever $P(\alpha n) > \alpha P(n)$ for a scaling factor $\alpha > 1$. This means that *operating a scalable SbN accrues an increasing profit margin*.

In what follows, we formally define the scalability of a social infiltration campaign from an economic perspective. After that, we develop the concepts needed to relate scalability to profitability.

Scalability in Economic Context

We generalize the scalability definition of online attacks presented by Herley [49], and adapt it to an arbitrarily-large social infiltration campaign. Formally, we call an SbN *scalable* if $C(n)$ grows more slowly than $R(n)$ as $n \to \infty$. In other words, when $C(n) = o(R(n))$ asymptotically, which is true if the following equality holds:

$$\lim_{n \to \infty} \frac{C(n)}{R(n)} = 0. \tag{2.2}$$

Otherwise, we call the SbN *non-scalable*. A scalable SbN is favorable as it has a desirable effect on profitability, where the following equality now holds:

$$\lim_{n\to\infty} \frac{P(n)}{R(n)} = 1, \tag{2.3}$$

which means that given a scalable SbN, $P(n)$ grows as fast as $R(n)$, or similarly, $P(n) \approx R(n)$ as $n \to \infty$, where $C(n)$ has a diminishing effect on profit.

To this end, in order to judge the scalability of a given SbN, we need to precisely define its total cost $C(n)$ and reward $R(n)$, after which we can test the scalability condition defined in Equation 2.2.

The Cost Structure

For an SbN with scale n and size m, let $C_s(n)$ and $C_o(n)$ be the setup and operation costs of the SbN, respectively. The *setup cost* is the cost incurred due to creating m fake accounts before operating the SbN, and the *operation cost* is the cost incurred due to running the SbN for a particular period of time. The operation cost includes a *rental cost* $C_r(n)$ for operating the SbN in a botnet, and a *detection cost* $C_d(n)$ for creating new fake accounts in place of detected ones. As described in the thread model, we assume the SbN is deployed as part of an existing botnet, with at least one compromised machine that is available. In other words, the rental cost is fixed $C_r(n) = \bar{r}$, where $\bar{r}$ is the average rental cost of a single machine in a botnet for the duration of the campaign. As for the detection cost $C_d(n)$, let $\bar{p}$ be the probability that a socialbot is detected during a single run of the campaign, we therefore have $C_d(n) = \bar{p} \times C_s(n)$. Accordingly, we define the total cost $C(n)$ as follows:

$$\begin{aligned} C(n) &= C_s(n) + C_o(n) \\ &= C_s(n) + C_r(n) + C_d(n) \\ &= C_s(n)(1+\bar{p}) + \bar{r}. \end{aligned} \tag{2.4}$$

As outlined in Section 2.5.2, each socialbot is limited to up to k social connec-

tions. This means $m = \lceil n/k \rceil$ bots are needed, and $C_s(n)$ is linearly dependent on n whenever $n > k$. Specifically, we define the setup cost $C_s(n)$ as follows:

$$C_s(n) = \begin{cases} \bar{c} & \text{if } n \leq k, \\ \bar{c} \times \left\lceil \frac{n}{k} \right\rceil & \text{if } n > k. \end{cases} \tag{2.5}$$

From Equations 2.4 and 2.5, the total cost $C(n) = O(1)$ for $n \leq k$, as the botherder needs to create a *fixed* amount of socialbots $m = \hat{m} = O(1)$. In this case, the following inequality holds for any scaling factor $\alpha > 1$, as long as $\alpha n \leq k$:

$$C(\alpha n) < \alpha C(n). \tag{2.6}$$

Accordingly, we say the SbN achieves an *economy of scale*: The cost of operating an SbN decreases per attacked user as the number of users to attack increases. For $n > k$, however, we have $C(n) = O(n)$ and the complement of Equation 2.6 holds for any scaling factor $\alpha > 1$:

$$C(\alpha n) \geq \alpha C(n). \tag{2.7}$$

In this case, each attacked user adds cost at least as much as the previous one, and the SbN has a growing size $m = O(n)$, which is defined by both n and k.

The Basic Reward Model

As mentioned in Section 2.5.2, there is no clear pricing structure for SbN goods. Therefore, we assume $R(n)$ grows at least linearly as $n \to \infty$, that is, $R(n) = \Omega(n)$. Moreover, this assumption is usually made upon entering new markets that do not offer short-term feedback [121], in which case the simplest way for valuation is to consider the average extracted value $\bar{v}$ and quote it for each unit of goods. Such a simple linear reward model, however, does not accommodate the *network effect*,

nor the market's pricing fluctuations or trends.[19] Accordingly, this model is useful for basic, short-term analysis, which is suitable for today's underground economy, especially in the presence of dishonesty, cheating, and uncertainty [50].

To make things concrete, let $\bar{y}$ be the yield of infiltration, which represents the average acceptance rate of a connection request sent by a socialbot. For now, we define the reward $R(n)$ of a social infiltration campaign with scale n as follows:

$$R(n) = n \times \bar{y} \times \bar{v} = O(n). \tag{2.8}$$

Scalability vs. Profitability

The definition of scalability (Equation 2.2) under the cost and reward definitions (Equations 2.5 and 2.8) leads to the following basic result: For $n \leq k$, the SbN is scalable and has a fixed size $\hat{m} = O(1)$, but for $n > k$, the SbN is non-scalable and has a growing size $m = O(n)$.

To give a meaningful interpretation of scalability, we now derive an economic implication of the result presented above. Let us consider the profit when we scale the SbN operation by a factor $\alpha > 1$. For $\alpha n \leq k$, we have:

$$\begin{aligned} P(\alpha n) &= R(\alpha n) - C(\alpha n) \\ &> \alpha R(n) - \alpha C(n) \\ &> \alpha P(n), \end{aligned} \tag{2.9}$$

which means that the botherder earns more profit by attacking, say, $100n$ users as opposed to n users when using a fixed number of socialbots. There is a clear economic incentive for the attacker to use the bots to go as large in scale as possible.

[19] The network effect is the effect a user of a product has on the value of the product to others. When present, the value of a product is dependent on the number of others using it [121].

When $\alpha n > k$, however, the situation changes as follows:

$$\begin{aligned} P(\alpha n) &= R(\alpha n) - C(\alpha n) \\ &\leq \alpha R(n) - \alpha C(n) \\ &\leq \alpha P(n), \end{aligned} \tag{2.10}$$

which means when operating an SbN using a growing number of socialbots, there is no benefit, if not a loss, from scaling the SbN and attacking even more users. The botherder makes the same profit, at best, if n or $100n$ users are attacked. This leads to the following non-desirable situation: The botherder has an incentive to scale the infiltration using a fixed number of socialbots, but has no incentives to infiltrate more users by operating a growing network of socialbots. In other words, for a rational botherder in a web of scale, social infiltration evidently suffers from the plight of non-scalability. In what follows, we derive two profit-maximizing social infiltration strategies which implement different business models under different scalability requirements.

2.5.4 Profit-Maximizing Infiltration Strategies

The main result from Section 2.5.3 is that *for a a social infiltration campaign to be scalable, it ought to use a fixed amount of socialbots*. In other words, the size of the SbN should be fixed $m = \hat{m} = O(1)$ while its scale should be as large as possible. However, scalable infiltration reaches only $\hat{m} \times k$ users. Ideally, one would like the reach to be unbounded for a scalable campaign, while being profitable.

In what follows, we show how the botherder can make profit by following a profit-maximizing infiltration strategy that implements one of two business models. In the first model, the botherder makes profit by selling harvested user data through infiltrating as many users as possible, following a scalable *data-driven* infiltration strategy. In the second model, the botherder makes profit by receiving a lump-sum payment through infiltrating a predefined number of users, following a non-scalable *target-driven* infiltration strategy.

Scalable Data-driven Infiltration

In order to achieve scalability, we need to upper bound the total cost $C(n)$ by $o(n)$ instead of $O(n)$ whenever $n > k$. The botherder can do so by artificially removing the limit k on each socialbot, and thus, making the cost independent from n. One way to achieve this is by breaking one or more social connections between each socialbot and attacked users after they become victims, or whenever the limit k is reached.[20] Doing so means the SbN can have a fixed size $\hat{m} = O(1)$, and therefore the total cost becomes independent from n as follows:

$$\begin{aligned} C(n) &= \bar{r} + C_s(n)(1+\bar{p}) \\ &= \bar{r} + (\bar{c} \times \hat{m})(1+\bar{p}). \end{aligned} \tag{2.11}$$

Accordingly, we have $C(\alpha n) < \alpha C(n)$ for a scaling factor $\alpha > 1$. In other words, the SbN is now scalable and achieves economy of scale—the botherder has clear incentives to scale its operation to infiltrate as many OSN users as possible.

We call this infiltration strategy *data-driven*, since the botherder seeks to harvest as much data from users as possible by following a simple infiltrate-collect-break strategy. In this strategy, however, the yield should be refined to include the average data access ratio per victim $\bar{d}$, as follows:

$$\hat{y} = \bar{y} \times \bar{d}. \tag{2.12}$$

We are interested in finding the scale n which maximizes the profit earned by operating the SbN under this data-driven infiltration strategy. Formally, we aim to solve the following profit-maximization problem:

$$\begin{aligned} \underset{n}{\text{maximize}} \quad & P(n) = \overbrace{n \times \bar{y} \times \bar{d} \times \bar{v}}^{R(n)} - \overbrace{\bar{r} + (\bar{c} \times \hat{m})(1+\bar{p})}^{C(n)} \\ \text{subject to} \quad & n \geq 1 \end{aligned}$$

[20] Recall that a socialbot collects user data immediately after an attacked user becomes a victim.

In order to maximize the profit, the botherder needs to maximize the reward or minimize the cost. As the total cost is fixed, the optimal solution is set n as large as possible, which means there is no closed form solution for n. Therefore, as $n \to \infty$, the profit $P(n) \to \infty$ as well, which is an expected result given the scalability of this infiltration strategy.

Non-Scalable Target-driven Infiltration

Another strategy is to keep the social connections with the victims, but to attack as many users as required in order to meet a contractually-agreed number of victims $\hat{n} \geq 1$, after which the botherder receives a lump sum payment $R(n) = \rho$ from a third-party. We call such a non-scalable strategy *target-driven*: The botherder operates the SbN on a limited scale to victimize $\hat{n}$ users on average. The scale and size of the SbN are derived by solving the following profit-maximization problem:

$$\begin{aligned} \underset{n,m}{\text{maximize}} \quad & P(n,m) = \overbrace{n \times \bar{y} \times \bar{v}}^{R(n)} - \overbrace{\bar{r} + (\bar{c} \times m)(1+\bar{p})}^{C(n)} \\ \text{subject to} \quad & m \geq \left\lceil \frac{n}{k} \right\rceil, n \geq \hat{n}, \text{ and } R(n) \leq \rho \end{aligned}$$

The optimal solution is find the minimal size m that is large enough to earn the lump-sum payment, in which the botherder is expected to victimize $\hat{n}$ users. By Equation 2.8, if the botherder attacks n users then the expected number of victims is $n \times \bar{y}$. Therefore, for a target of $\hat{n}$ victims, the optimal solution is to set the scale to $n = \lceil \hat{n}/\bar{y} \rceil$ and the size to $m = \lceil n/k \rceil = \lceil \lceil \hat{n}/\bar{y} \rceil / k \rceil$.

2.5.5 Case Study: Social Infiltration in Facebook

We next assume the role of a botherder who plans to run a social infiltration campaign in Facebook. In particular, we apply the analysis from Section 2.5.4 to decide whether it is economically feasibility for the botherder to operate an SbN under each one of the profit-maximizing infiltration strategies.

Parameter	Description	Estimate	Source
$\bar{p}$	Probability of detecting a fake account	0.2	Boshmaf et al. [10]
$\bar{y}$	Average friend request acceptance rate	0.536	Boshmaf et al. [10]
$\bar{r}$	Average rental cost of a machine in a botnet	\$0.35	Goncharov [41]
$\bar{c}$	Average cost of creating a fake account	\$0.1	Thomas et al. [114]
$\bar{d}$	Average email address access ratio	0.718	Boshmaf et al. [10]
$\bar{v}$	Average extracted value per email address	¢0.03	Fossi et al. [37]
k	Maximum number of friends per user	5,000	Boshmaf et al. [10]
$\hat{n}$	Number of users to befriend and victimize	38,462	Motoyama et al. [88]
ρ	Lump-sum payment for befriending $\hat{n}$ users	\$1,000	Motoyama et al. [88]

Table 2.3: Estimates for analyzing infiltration profitability in Facebook

Campaign Setup

Let us consider a botherder who wants to operate an SbN constructed as described in Section 2.4. Being rationale, the bothered wants to decide whether it would be economically feasible to operate the SbN in the first place. The botherder targets Facebook, where the average acceptance rate of a friend request is $\bar{y} = 0.536$ after seeding. There is also a probability $\bar{p} = 0.2$ for each fake account to be detected.

To operate the SbN, the botherder needs to create fake accounts and rent at least one machine in a botnet for the duration of the campaign. Thomas et al. [114] reported that 1K phone-verified fake accounts costs \$100 ($\bar{c} = \$0.1$), where merchants like BuyAccs[21] making millions of dollars in revenue each year. Moreover, a recent study [41] by TrendMicro on Russian underground markets reported that the average rental cost of 2K hijacked machines in a botnet is \$200 ($\bar{r} = \$0.35$), while buying a similar botnet costs as little as \$700. Table 2.3 provides a summary of these and other campaign parameters.

Economic Feasibility

In what follows, we apply the strategies derived in Section 2.5.4 in practice, and show that operating an SbN on Facebook at scale is expected to be profitable but

[21] http://buyaccs.com

is not financially attractive as an independent business.

Collecting and Selling Email Addresses

Let us start with a data-driven infiltration campaign where the borherder operates $\hat{m} = 100$ socialbots to collect and sell email addresses at underground markets.[22]

As presented in Section 2.4, the average data access ratio of email addresses is $\bar{d} = 0.718$ after infiltration. A recent study [37] by Symantec on underground markets reported that 1MB of email addresses is worth \$0.3–\$40 at IRC underground markets, with an average of \$20.15 per MB. Assuming an email address is 15 ASCII characters long, or 15 bytes, then an email address has an average value of $\bar{v} = ¢0.03$. A more recent study [115] by Fortinet about the anatomy of botnets reported a similar value of \$100 for a million email addresses.

We can now calculate the profit in dollars for the SbN with scale n by substituting the values from Table 2.3 into Equations 2.1, 2.11, and 2.12, leading to:

$$\begin{aligned} P(n) &= R(n) - C(n) \\ &= \$0.0001154544n - \$12.35. \end{aligned}$$

The botherder breaks even after attacking $n_0 \approx 107$K users. To make \$1K in profit, the botherder has to attack nearly 8.7 million users. In fact, the botherder can earn at most \$155,851 by attacking each one of the 1.35 billion users on Facebook [30]. While unrealistic, this figure represents an optimistic estimate of how much user data, email addresses in particular, are worth in underground markets, even for the unlikely case when the botherder manages to run such an extreme-scale campaign.

The main result here is that *running a large-scale social infiltration campaign is not expected to incur high cost nor high profits*. It facilitates, however, a non-zero profit monetization campaign for more profitable underground commodities.

[22] We chose to restrict the collected data to email addresses in order to avoid over-estimating the extracted value of user profile information.

Befriending Users for Profit

We now consider a target-driven infiltration campaign, in which the figures look more promising for the botherder. Motoyama et al. [88] recently reported that one can earn $\rho = \$1,000$ for befriending $\hat{n} = 38,462$ users in online freelance markets such as Freelancer.[23] The authors refer to this task as *OSN linking*, where a freelance worker creates enough accounts in the target OSN to befriend a particular number of users, after which the worker receives an specific lump-sum payment. Given the limit of $k = 5,000$ friends per user account on Facebook [10], the botherder attacks $n = \lceil \hat{n}/\bar{y} \rceil = 71,758$ users using $m = \lceil n/k \rceil = 15$ socialbots. By directly applying Equations 2.1 and 2.5, where $R(n) = \rho$, the botherder is expected to make a profit of $P(71,758) = \$1,000 - \$2.15 = \$997.85$.

To this end, *the botherder can make more profit by operating a non-scalable social infiltration campaign*. As compared to the earlier strategy, the botherder attacks nearly 120 times less users and makes about the same profit of $1K, which makes the latter strategy more financially attractive. In fact, Motoyama et al. [88] found that particular freelance tasks, such as OSN linking and creating fake accounts, were finished in less than a minute, suggesting the use of automation software to undertake these non-scalable tasks.

2.5.6 Discussion

We now discuss the implications of social infiltration on underground market. In particular, we make the case that even if a social infiltration campaign is profitable, it is still more reasonable for a botherder to treat it as a monetization campaign for more profitable commodities, rather than an independent business by itself.

Implications for Underground Markets

Using a scalable data-driven SbN means its operation ought to be automated and non-adaptive to individual users, while delivering commodity-goods as a result.

[23] http://freelancer.com/

These implications are attributed to the economy of scale the strategy achieves, and are discussed in depth by Herely [49]. In particular, adding per-user personalization, such as responding to individual messages requesting introductions, or adapting to per-user countermeasures, such as breaking CAPTCHAs to send more connection requests, violates the cost structure of the strategy and would result in potential losses. Moreover, as the operation of the SbN is automated, other copycat attackers will have the incentives to operate a similar SbN as well. As Herley nicely puts it [49]: "once a clever exploit is scripted, then it can be used by many." Consequently, this automation will increase the demand for large-scale social infiltration campaigns but decrease their profit, resembling the effect of the *tragedy of commons*: A dilemma arising from the situation in which multiple individuals, acting independently and rationally, will ultimately deplete a shared limited resource, even when it is clear that it is not in anyone's long-term interest [121]. This has the implication of decreasing the average extracted value $\bar{v}$ down to zero.

In a non-scalable target-driven SbN, the situation is different. As the objective of the botherder is to reach a target number of victims rather than to attack users at scale to collect their data, it is more reasonable to target users who are expected to have a higher yield $\bar{y}$. As shown in Section 2.4, such users have a higher than average number of friends on Facebook. For this the botherder is fortunate, the *degree distribution* of users in OSNs is power-law and highly concentrated [80]. However, this also means the botherder will ignore the majority of users because only a small fraction of users have higher than average number of friends.

Scalable Social Infiltration as a Business

Even though the profit estimates in Section 2.5.5 might be optimistic, it is safe to assume that operating a data-driven SbN is expected to make a non-zero profit. In what follows, we argue that such an SbN is not attractive as an independent and sustainable business, and a rational botherder would utilize the SbN as a monetization tool for subsequent, more profitable campaigns.

First, the analysis we presented in Section 2.4.5 is a cost-volume-profit (CVP)

analysis, which is useful for short-run decisions and has to be re-evaluated regularly. Even if the botherder is able to observe market trends over time—which allows planning a pricing structure, using more predictive economic models, and forecasting future profits—the underground market is still unfriendly. Herley et al. [50] used simple economic arguments to show that such markets suffer from cheating, dishonesty, and uncertainty. Also, the authors showed that these markets represent a classic example of *markets for lemons* [121]: The situation where sellers have better information than buyers about the quality of their goods, which leads to an adverse selection where the bad drives out the good.[24] These markets drive "high-quality" businesses out of existence and result in a diminishing valuation of goods. After all, "nobody sells gold for the price of silver." [50]

Second, maintaining a business requires durability. Operating an SbN at scale, however, is expected to become more costly to maintain over time. This is due to the fact that OSN operators update their deployed defenses to mitigate online attacks, especially against large-scale ones. With *adversarial learning* systems in place, such as Facebook's immune system [104], this will result in an arms race between the OSN operator and the botherder, where the more resourceful party will eventually win. Based only on the profit the botherder is expected to make from operating a data-driven SbN, the odds are small that the investment in maintaining and updating the SbN will ever payback, especially as this would force the SbN to become non-scalable due to the increased cost. As demonstrated by Florêncio et al. [35], the botherder faces a sum-of-effort rather than a weakest-link defense, which means even a slight improvement in defenses employed by the OSN or its users can render the SbN non-scalable and non-profitable.

Finally, Kanich et al. [61] showed that the underground market for spam-advertised businesses is particularly attractive, where such businesses make hundreds of thousands of dollars a month in revenue. The botherder is thus better off using an SbN as a tool for monetizing subsequent, personalized e-mail spam cam-

[24] A *lemon* is an American slang term for a car that is found to be defective only after it has been bought. The market for lemons concludes that owners of good cars will not place their cars on the used-car market. This is sometimes summarized as "the bad driving out the good" in the market.

paign as part of a larger underground affiliation. In other words, large-scale social infiltration can be considered as a market enabler of the actual "money-makers," as confirmed by other empirical studies on Twitter [109, 114].

2.6 Summary

From a computer security perspective, the concept of socialbots is both interesting and disturbing. The threat is no longer from a human controlling or monitoring a computer, but from exactly the opposite.

This study presented an empirical evidence showing that OSNs such as Facebook are vulnerable to social infiltration by socialbots—automated fake accounts that mimic real user behavior. In particular, we found that such socialbots make it difficult for OSN security defenses, such as Facebook's immune system, to detect or stop social infiltration as it occurs. This has resulted in alarming privacy breaches and serious implications to graph-based fake account detection mechanisms, which assume attackers who are not capable of social infiltration.

We also analyzed the economic feasibility of social infiltration at scale. The analysis suggested that large-scale social infiltration, when operated as part of a socialbot network (SbN), is expected to be profitable but is not particularly attractive as an independent business. Social infiltration at scale, however, plays the role of a market enabler of more profitable commodities, which are monetized through social spam and malware in a multi-million dollar underground market. In addition, we showed that for an SbN to be scalable in terms of number of attacked users, it ought to have a fixed size in terms of number of its socialbots. Otherwise, there would be a linear cost dependence which forces the botherder to limit the scale of the SbN in order for it to be more financially attractive.

While this study showed that social infiltration in OSNs is both profitable and difficult to detect, it has also made the observation that victims of socialbots have predictive characteristics. As shown in Chapter 3, this insight has practical implications to the design of robust defense mechanisms that lead to a safer Web for billions of active OSN users.

Chapter 3

Infiltration-Resilient Fake Account Detection in OSNs

In its 2014 earnings report, Facebook estimated that 15–16 millions (1.2%) of its monthly active users are in fact "undesirable," representing fake accounts that are used in violation of the site's terms of service [32]. For such OSNs, the existence of fakes leads advertisers, developers, and investors to distrust their reported user metrics, which negatively impacts their revenues [17]. Moreover, as we discussed in Chapter 2, attackers create and automate fake accounts, or socialbots, for various malicious activities, starting with social infiltration. Therefore, it is important for OSNs to detect fake accounts as fast and accurately as possible.

Most OSNs employ defense systems that automatically flag fake accounts by analyzing either user-level activities or graph-level structures. Because automated account suspension is inapplicable in practice, these accounts are pooled for manual verification by experienced analysts, who maintain a ground-truth for fake and real accounts [15, 104].

Fake account detection, in general, can be divided into two main approaches. In the first approach, unique *features* are extracted from recent user activities, such as frequency of friend requests and fraction of accepted requests, after which they are applied to a fake account classifier that has been trained offline using machine

learning techniques [104]. In the second approach, an OSN is modeled as a *graph*, with nodes representing user accounts and edges representing social relationships (e.g., friendships). Given the assumption that *fake accounts can befriend only few real accounts*, the graph is partitioned into two subgraphs, or *regions*, separating real accounts from fakes, with a narrow passage between them [2, 137]. While both approaches are effective against naïve attacks, various studies showed that they are inaccurate in practice and can be easily evaded [6, 10, 29, 125]. In Chapter 2, we showed that an attacker can cheaply create fake accounts that resemble real users, circumventing feature-based detection, or use simple social engineering tactics to befriend and infiltrate many real users, invalidating the assumption behind graph-based detection (Finding 4).

In this chapter, we consider attackers who can run a large-scale social infiltration campaign using a set of automated fake accounts, or socialbots. Specifically, each fake account can perform social activities similar to those of real users, including befriending other users. Under this stronger threat model, we aim to tackle the following question in our design of Íntegro, as introduced in Section 1.3.2:

- **RQ4:** How can OSNs detect fakes that infiltrate users on a large scale?

3.1 Background and Related Work

We first review the threat model we assume in this work. We then present required background and related work on fake account detection, abuse mitigation, maintaining a ground-truth, and analyzing victim accounts in OSNs.

3.1.1 Threat Model

We focus on large OSNs such as Tuenti, RenRen, and Facebook, which are open to everyone and allow users to declare bilateral relationships (i.e., friendships).

Capabilities

As presented in Chapter 2, we consider attackers who are capable of creating and automating fake accounts on a large scale. Each fake account, also referred to as a socialbot, can perform social activities similar to those of real users [56]. This includes sending friend requests and posting social content. We do not consider attackers who are capable of hijacking real accounts, as there are existing detection systems that tackle this threat, such as COMPA [26]. We focus on detecting fake accounts that can befriend a large number of benign users in order to mount subsequent attacks, as we describe next.

Objectives

The objectives of an attacker include distributing social spam and malware, misinforming the public, and collecting private user data on a large scale. To achieve these objectives, the attacker has to *infiltrate* the target OSN by using the fakes to befriend many real accounts. Such an infiltration is required because isolated fake accounts cannot directly interact with or promote content to most users in the OSN [28]. This is also evident by a thriving underground market for OSN abuse, including social infiltration. For example, attackers can have their fake accounts befriend 1K users on Facebook for $26 or less [88].

Victims

We refer to accounts whose users have accepted friend requests sent by fake accounts as *victims*. We refer to friendships between victim and fake accounts as *attack edges*. Victim accounts are subset of *real* accounts, which are accounts created and controlled by benign users who socialize with others in a non-adversarial setting. Moreover, we refer to accounts whose users are more susceptible to social infiltration and are likely to be victims as *potential victims*. We use the terms "account," "profile," and "user" interchangeably but do make the distinction when deemed necessary.

3.1.2 Fake Account Detection

From a systems design perspective, most of today's fake account detection mechanisms are either feature-based or graph-based, depending on whether they utilize machine learning or graph analysis techniques in order to identify fakes. Next, we discuss each of these approaches in detail.

Feature-based Detection

This approach relies on user-level activities and its account details (i.e., user logs, profile pages). By identifying discriminating *features* of an account, one can classify each account as fake or real using various machine learning techniques. For example, Facebook employs an "immune system" that performs real-time checks and classification for each read and write action on its database, which are based on features extracted from user accounts and their activities [104].

Yang et al. used ground-truth provided by RenRen to train an SVM classifier in order to detect fake accounts [134]. Using simple features, such as frequency of friend requests, fraction of accepted requests, and per-account clustering coefficient, the authors were able to train a classifier with 99% true-positive rate (TPR) and 0.7% false-positive rate (FPR).

Stringhini et al. utilized honeypot accounts in order to collect data describing various user activities in OSNs [108]. After analyzing the collected data, they were able to assemble a ground-truth for real and fake accounts, with features similar to those outlined before. The authors trained two random forests (RF) classifiers to detect fakes in Facebook and Twitter, ending up with 2% FPR and 1% false-negative rate (FNR) for Facebook, and 2.5% FPR and 3% FNR for Twitter.

Wang et al. used a click-stream dataset provided by RenRen to cluster user accounts into "similar" behavioral groups, corresponding to real or fake accounts [127]. The authors extracted both sessions and clicks features, including average clicks per session, average session length, the percentage of clicks used to send friend requests, visit photos, and to share content. With these features, the authors were able to calibrate a cluster-based classifier with 3% FPR and 1% FNR, using the

METIS clustering algorithm [64].

Cao et al. observed that fake accounts tend to perform loosely synchronized actions in a variety of OSN applications, all from a limited set of IP addresses [16]. The authors extracted simple user action features, such as timestamp, target application, and IP address, in order to cluster user accounts according to the similarity of their actions using a scalable implementation of the single-linkage hierarchical clustering algorithm. Through a deployment at Facebook, the authors we able to calibrate a cluster-based classifier and detect more than two million fake accounts, which acted similarly at about the same time for a sustained period of time.

Even though feature-based detection scales to large OSNs, it is still relatively easy to circumvent. This is the case as it depends on features describing activities of known fakes in order to identify unknown ones. In other words, attackers can evade detection by adversely modifying the content and activity patterns of their fakes, which leads to an arms race [74, 120]. In addition, feature-based detection does not provide any formal security guarantees and often results in a high FPR in practice. This is partly attributed to the large variety and unpredictability of behaviors of users in adversarial settings [15].

With Íntegro, we *use feature-based detection to identify potential victims in a non-adversarial setting*. In particular, the dataset used for training a victim classifier includes features of only known real accounts that have either accepted or rejected friend requests send by known fakes. Because real accounts are controlled by benign users who are not adversarial, a feature-based victim account classifier is much harder to circumvent than a similarly-trained fake account classifier. As we discuss in Section 3.3, we require the victim classification to be better than random in order to outperform the state-of-the-art in fake account detection.

Graph-based Detection

As a response to the lack of formal security guarantees in feature-based detection, the state-of-the-art in fake account detection utilizes a graph-based approach instead. In this approach, an OSN is modeled as a *graph*, with nodes representing

user accounts and edges between nodes representing social relationship. Given the assumption that fakes can establish only a small number of attack edges, the subgraph induced by the set of real accounts is sparsely connected to fakes, that is, the *cut* over attack edges is sparse.[1] Graph-based detection mechanisms make this assumption, and attempt to find such a sparse cut with formal guarantees [123, 137]. For example, Tuenti employs SybilRank to rank accounts according to their perceived likelihood of being fake, based on structural properties of its graph [15].

Yu et al. were among the first to analyze the social graph for the purpose of identifying fake accounts in OSNs [138, 139]. The authors developed a technique that labels each account as either fake or real based on multiple, modified random walks. This binary classification is used to partition the graph into two smaller subgraphs that are sparsely interconnected via attack edges, separating real accounts from fakes. They also proved that in the worst case $O(|E_a| \log n)$ fakes can be misclassified, where $|E_a|$ is the number of attack edges and n is the number of accounts in the network. Accordingly, it is sufficient for the attacker to establish $\Omega(n/\log n)$ attack edges in order to evade this detection scheme with 0% TPR.

Viswanath et al. employed community detection techniques to identify fake accounts in OSNs [122]. In general, community detection decomposes a given graph into a number of tightly-knit subgraphs that are loosely connected to each other, where each subgraph is called a *community* [36, 72]. By expanding a community starting with known real accounts [82], the authors were able to identify the subgraph which contains mostly real accounts. Recently, however, Alvisi et al. showed that such a local community detection technique can be easily circumvented if fake accounts establish sparse connectivity among themselves [2].

As binary classification often leads to high FPR [122], Cao et al. proposed user ranking instead, such that almost all fake accounts are ranked lower than real accounts [15]. The authors developed SybilRank, a fake account detection system that assigns each account a rank describing how likely it is to be fake based on a

[1] A cut is a partition of nodes into two disjoint subsets. Visually, it is a line that cuts through or crosses over a set of edges in the graph (see Figure 3.1).

modified random walk, in which a lower rank means the account is more likely to be fake. They also proved that $O(|E_a| \log n)$ fakes can outrank real accounts in the worst case, given the fakes establish $|E_a|$ attack edges with victims at random.

While graph-based detection provides provable security guarantees, real-world social graphs do not conform with the main assumption on which it depends. In particular, various studies confirmed that attackers can infiltrate OSNs on a large scale by deceiving users into befriending their fakes [6, 29, 125], which renders graph-based fake account detection ineffective in practice.

With Íntegro, we *do not assume that fake accounts are limited by how many attack edges they can establish*. Instead, we identify potential victims and leverage this information to carefully weight the graph. After that, through a user ranking scheme, we bound the security guarantee by the aggregate weight on attack edges, $\text{vol}(E_a)$, rather than their number, $|E_a|$. In particular, by assigning lower weights to edges incident to potential victims than other accounts, we can upper bound the value of $\text{vol}(E_a)$ by $|E_a|$, as we show in Section 3.3.

3.1.3 Abuse Mitigation and the Ground-truth

Due to the inapplicability of automated account suspension, OSNs employ abuse mitigation techniques, such as CAPTCHA challenges [104] and photo-based social authentication [136], in order to rate-limit accounts that have been automatically flagged as fake. These accounts are pooled for manual inspection by experienced analysts who maintain a ground-truth for real and fake accounts along with their features, before suspending or removing verified fakes [15, 104, 128, 134].

While maintaining an up-to-date ground-truth is important for retraining deployed classifiers and estimating their effectiveness in practice, it is rather a time-consuming and non-scalable task. For example, on an average day, each analyst at Tuenti inspects 250–350 accounts an hour, and for a team of 14 employees, up to 30K accounts are inspected per day [15]. It is thus *important to rank user accounts in terms of how likely they are to be fake in order to prioritize account inspection by analysts*. Íntegro offers this functionality and leads to a faster reaction against

potential abuse by fakes, benefiting both OSN operators and their users.

3.1.4 Analyzing Victim Accounts

While we are the first to leverage victim classification to separate fakes from real accounts in the graph, other researchers have analyzed victim accounts as part of the larger cyber criminal ecosystem in OSNs [109].

Wagner et al. were among the first to develop predictive models in order to identify users who are more susceptible to social infiltration in Twitter [125]. They found that susceptible users, or potential victims, tend to use Twitter for conversational purposes, are more open and social since they communicate with many different users, use more socially welcoming words, and show higher affection than non-susceptible users.

Yang et al. studied the cyber criminal ecosystem on Twitter [133]. They found that victims fall into one of three categories. The first are *social butterflies* who have large numbers of followers and followings, and establish social relationships with other accounts without careful examination. The second are *social promoters* who have large following-follower ratios, larger following numbers, and a relatively high URL ratios in their tweets. These victims use Twitter to promote themselves or their business by actively following other accounts without consideration. The last are *dummies* who post few tweets but have many followers. In fact, these victims are dormant fake accounts at an early stage of their abuse.

3.2 Intuition, Goals, and Model

We now introduce Íntegro, a fake account detection system that is *robust against social infiltration*. In what follows, we first present the intuition behind our design, followed by its goals and model.

3.2.1 Intuition

We start with the premise that some users are more likely to be victims than others. If we can train a classifier to identify potential victims with high probability, we may be able to use this information to find the cut which separates fakes from real accounts in the graph. As victims are benign users, the output of this classifier represents a reliable information which we can integrate in the graph.

To find such a cut, we can define a graph weighting scheme that assigns edges incident to potential victims lower weights than others, where weight values are calculated from classification probabilities. In a weighted graph, the sparsest cut is the cut with the smallest *volume*, which is the sum of weights on edges across the cut. Given an accurate victim classifier, this cut is expected to cross over some or all attack edges, effectively separating real accounts from fakes, even if the number of attack edges is large. We aim to find such a cut using a ranking scheme that ideally assigns higher ranks to nodes in one side of the cut than the other, as one way to separate real accounts from fakes. This ranking scheme is inspired by similar graph partitioning algorithms proposed by Spielman et al. [103], Yu [137], and Cao et al. [15].

3.2.2 Design Goals

Íntegro is designed to help OSNs detect fake accounts, which are capable of large-scale social infiltration, through a user ranking scheme. In particular, Íntegro has the following design goals:

Effectiveness: High-Quality User Ranking

The system should ideally assign higher ranks to real accounts than fakes. If not, it should limit the number of fakes that might rank similar to or higher than real accounts. In practice, the ranking should *have an area under ROC curve (ROC) that is greater than 0.5 and closer to 1*, where the AUC represents the probability of a random real accounts to rank higher than a random fake account [122].

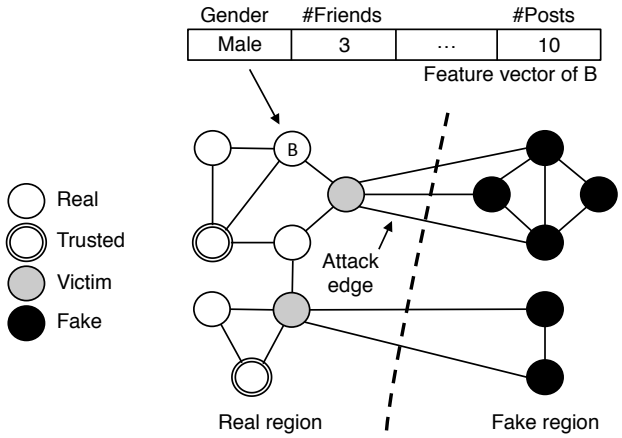


Figure 3.1: System model

Also, the system should be robust against social infiltration under real-world attack strategies. Given a ranked list of users, a high percentage of the users at the bottom of the list should be fake. This percentage, which represents the *precision* of detection, should significantly decrease as we move to higher ranks in the list.

Efficiency: Scalability and Easy Deployment

The system should have a practical computational cost that allows it to scale to large OSNs. In other words, it should *scale nearly linearly with number of user accounts* in the OSN, and deliver ranking results in only few minutes. The system should be able to extract useful, low-cost features and process large graphs on commodity machines, so that OSN operators can deploy it on their existing computer clusters.

3.2.3 System Model

As illustrated in Figure 3.1, we model an OSN as an undirected graph $G = (V, E)$, where each node $v_i \in V$ represents a user account and each edge $\{v_i, v_j\} \in E$ represents a bilateral social relationship among v_i and v_j. In the graph G, there are $n = |V|$ nodes and $m = |E|$ edges.

Attributes

Each node v_i has a *degree* $\deg(v_i)$, which is equal to the sum of weights on edges incident to the node. In addition, v_i has a *feature vector* $\mathcal{A}(v_i)$, where each entry $a_j \in \mathcal{A}(v_i)$ describes a feature or an attribute of the account. Each edge $\{v_i, v_j\} \in E$ has a *weight* $w(v_i, v_j) \in (0, 1]$, which is initially set to 1.

Regions

The node set V is divided into two disjoint sets, V_r and V_f, representing real and fake accounts, respectively. We refer to the subgraph induced by V_r as the *real region* G_r, which includes all real accounts and the friendships between them. Likewise, we refer to the subgraph induced by V_f as the *fake region* G_f. The regions are connected by a set of attack edges E_a between victim and fake accounts. We assume the OSN operator is aware of a small set of *trusted accounts* V_t, also called *seeds*, which are known to be real accounts and are not victims.

3.3 System Design

We now describe the design of Íntegro. We start with a short overview of our approach, after which we proceed with a detailed description of system components.

3.3.1 Overview

In the beginning, Íntegro trains a *victim classifier* using low-cost features extracted from user-level activities. This feature-based binary classifier is used to identify potential victims in the graph, each with some probability (Section 3.3.2). After that, Íntegro calculates new edge weights from the probabilities computed by the victim classifier in such a way that edges incident to potential victims have lower weights than others. Íntegro then ranks user accounts based on the landing probability of a modified random walk that starts from a trusted account, or a seed node, picked at random from all trusted accounts (Section 3.3.3).

The random walk is "short" because it is terminated after $O(\log n)$ steps, early

before it converges. The walk is "supervised," as it is biased towards traversing nodes which are reachable via higher-weight paths. This modified random walk has a higher probability to stay in the real region of the graph, because it is highly unlikely to escape into the fake region in few steps through low-weight attack edges. As a result, Íntegro ranks most of real accounts higher than fakes, given a seed selection strategy that considers the existing community structures in the real region (Section 3.3.4).

Íntegro takes $O(n \log n)$ time to complete its computation (Section 3.3.5). In addition, it formally guarantees that at most $O(\text{vol}(E_a) \log n)$ fake accounts can have the same or higher ranks than real accounts in the worst case, given the fakes establish $|E_a|$ attack edges at random (Section 3.3.6).

3.3.2 Identifying Potential Victims

For each user v_i, Íntegro extracts a feature vector $\mathcal{A}(v_i)$ from its user-level activities. A subset of the feature vectors is selected to train a binary classifier in order to identify potential victims using supervised machine learning, as follows:

Feature Engineering

Extracting and selecting useful features from user-level activities can be a challenging and time consuming task. To be efficient, we seek *low-cost* features which could be extracted in $O(1)$ time per user account. One possible location for extracting such features is the profile page of user accounts, where features are readily available (e.g., a Facebook profile page). However, low-cost features are expected to be *statistically weak*, which means they may not strongly correlate with the *label* of a user account (i.e., victim or not). As we explain later, we require the victim classifier to be better than random in order to deliver robust fake account detection. This requirement, fortunately, is easy to satisfy. In particular, we show in Section 3.4 that an OSN can train and cross-validate a victim classifier that is up to 52% better than random, using strictly low-cost features.

Supervised Machine Learning

For each user v_i, Íntegro computes a *vulnerability score* $p(v_i) \in (0,1)$ that represents the probability of v_i to be a victim. Given an *operating threshold* $\alpha \in (0,1)$ with a default value $\alpha = 0.5$, we say v_i is a potential victim if $p(v_i) \geq \alpha$. To compute vulnerability scores, Íntegro uses random forests (RF) learning algorithm [12] in order to train a victim classifier, which given $\mathcal{A}(v_i)$ and α, decides whether the user v_i is a potential victim with a vulnerability score $p(v_i)$. We picked the RF learning algorithm because it is both efficient and robust against model over-fitting [47]. Íntegro takes $O(n \log n)$ time to extract n feature vectors and train an RF-based victim classifier. It also takes $O(n)$ to compute vulnerability scores for all users, given their feature vectors and the trained victim classifier.

Robustness

As attackers do not control victims, a victim classifier is inherently more resilient to adversarial attacks than similarly-trained fake account classifier. Let us consider one concrete example. In the "boiling-frog" attack [120], fake accounts can force a classifier to tolerate abusive activities by slowly introducing similar activities to the OSN. Because the OSN operator has to retrain all deployed classifiers in order to capture new behaviors, a fake account classifier will learn to tolerate more and more abusive activities, up until the attacker can launch a full-scale attack without detection [10]. When identifying potential victims, however, this is only possible if the real accounts used to train the victim classifier have been compromised. This situation can be avoided by verifying the accounts, as described in Section 3.1.3.

Generalization

While machine learning-based victim classification is imperfect and will typically result in false positives, it is highly scalable and requires only a small ground-truth to generalize the classification to the entire user-base, as we show in Section 3.4.5. This ability of a methodically trained and validated classifier to *generalize*, so that it performs accurately on unseen before feature vectors, is key for the predictive

power of machine learning [47]. If not, an OSN has to collect the missing ground-truth by directly testing whether each user is likely to be a victim, possibly via a benign social infiltration campaign that is run against the entire user-base. Such a brute-force approach to identifying potential victims is infeasible for three main reasons. First, it has a considerably high cost to the OSN and its users, as it needs to be administered regularly against all accounts to capture changing user behaviors. Second, flooding the network with friend requests at random is highly inefficient, as only a small fraction of users are expected to become victims. Third and last, forcing users to participate in the test introduces its own ethical concerns [9], as it could result in risky user behaviors such as emotional attachment, in addition to the corresponding negative impact on user experience.

3.3.3 Ranking User Accounts

To rank users, Íntegro computes the probability of a modified random walk to land on each user v_i after k steps, where the walk starts from a trusted user account picked at random. For simplicity, we refer to the probability of a random walk to land on a node as its *trust value*, so the probability distribution of the walk at each step can be modeled as a *trust propagation process* [45]. In this process, a weight $w(v_i, v_j)$ represents the rate at which trust may propagate from either side of the edge $\{v_i, v_j\} \in E$. We next describe this process in detail.

Trust Propagation

Íntegro utilizes the *power iteration method* to efficiently compute trust values [40]. This method involves successive matrix multiplications where each element of the matrix is the transition probability of the random walk from one node to another. Each iteration computes the trust distribution over nodes as the random walk proceeds by one step. Let $T_k(v_i)$ denote the trust collected by each node $v_i \in V$ after k iterations. Initially, the *total trust*, denoted by $\tau \geq 1$, is evenly distributed among

the trusted nodes in V_t:

$$T_0(v_i) = \begin{cases} \tau/|V_t| & \text{if } v_i \in V_t, \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

The process then proceeds as follows:

$$T_k(v_i) = \sum_{\{v_i,v_j\}\in E} T_{k-1}(v_j) \cdot \frac{w(v_i,v_j)}{\deg(v_j)}, \tag{3.2}$$

where in iteration k, each node v_i propagates its trust $T_{k-1}(v_i)$ from iteration $k-1$ to each neighbour v_j, proportionally to the ratio $w(v_i,v_j)/\deg(v_i)$. This is required so that the sum of the propagated trust equals $T_{k-1}(v_i)$. The node v_i then collects the trust propagated similarly from each neighbour v_j and updates its trust $T_k(v_i)$. Throughout this process, τ is preserved such that for each iteration $k \geq 1$ we have:

$$\sum_{v_i\in V} T_{k-1}(v_i) = \sum_{v_i\in V} T_k(v_i) = \tau. \tag{3.3}$$

Our goal here is to ensure that most real accounts collect higher trust than fake accounts. In other words, we want to strictly limit the portion of τ that escapes the real region G_r and enters the fake region G_f. To achieve this propagation property, we make the following modifications.

Adjusted Propagation Rates

In each iteration k, the aggregate rate at which τ may enter G_f is strictly limited by the sum of weights on the attack edges, which we denote by the *volume* $\text{vol}(E_a)$. Therefore, we aim to adjust the weights in the graph such that $\text{vol}(E_a) \in (0, |E_a|]$, without severely restricting trust propagation in G_r. We accomplish this by assigning smaller weights to edges incident to potential victims than other accounts. In particular, each edge $\{v_i,v_j\} \in E$ keeps the default weight $w(v_i,v_j) = 1$ if v_i and

v_j are both *not* potential victims. Otherwise, we modify the weight as follows:

$$w(v_i, v_j) = \min\left\{1, \beta \cdot \left(1 - \max\{p(v_i), p(v_j)\}\right)\right\}, \tag{3.4}$$

where β is a scaling parameter with a default value of $\beta = 2$. Now, as $\text{vol}(E_a) \to 0$ the portion of τ that enters G_f reaches zero as desired.

For proper degree normalization, we introduce a self-loop $\{v_i, v_i\}$ with weight $w(v_i, v_i) = (1 - \deg(v_i))/2$ whenever $\deg(v_i) < 1$. Notice that self-loops are considered twice in degree calculation.

Early Termination

In each iteration k, the *trust vector* $T_k(V) = \langle T_k(v_1), \ldots, T_k(v_n)\rangle$ describes the distribution of τ throughout the graph. As $k \to \infty$, the vector converges to a stationary distribution $T_\infty(V)$, as follows [5]:

$$T_\infty(V) = \left\langle \tau \cdot \frac{\deg(v_1)}{\text{vol}(V)}, \ldots, \tau \cdot \frac{\deg(v_n)}{\text{vol}(V)} \right\rangle, \tag{3.5}$$

where the node-set volume $\text{vol}(V)$ in this case is the sum of degrees of nodes in V. In particular, $T_k(V)$ converges after k reaches the *mixing time* of the graph, which has been shown to be much larger than $O(\log n)$ iterations for various kinds of social networks [21, 72, 83]. Accordingly, we terminate the propagation process early before it converges after $\omega = O(\log n)$ iterations.

Degree Normalization

As described in Equation 3.5, trust propagation is influenced by individual node degrees. As k grows large, the propagation is expected to bias towards high degree nodes. This implies that high degree fake accounts may collect more trust than low degree real accounts, which is undesirable for effective user ranking. To eliminate this node degree bias, we normalize the trust collected by each node by its degree. We assign each node $v_i \in V$ after $\omega = O(\log n)$ iterations a *rank value* $T'_\omega(v_i)$ that

is equal to its degree-normalized trust:

$$T'_{\omega}(v_i) = T_{\omega}(v_i)/\deg(v_i). \tag{3.6}$$

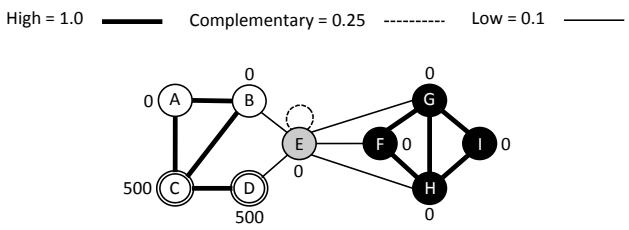
Finally, we sort the nodes by their ranks in a descending order.

Example

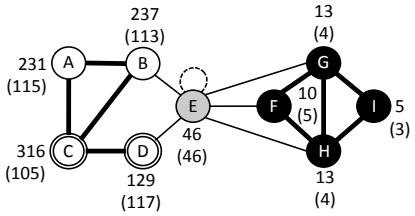
Figure 3.2 depicts trust propagation in a graph. In this example, we assume each account has a vulnerability score of 0.05, except the victim E, which has a score of $p(E) = 0.95$. The graph is weighted using $\alpha = 0.5$ and $\beta = 2$, and a total trust $\tau = 1000$ in initialized over the trusted nodes $\{C, D\}$. Each value is rounded to its nearest natural number. The values with parentheses represent the corresponding degree-normalized trust (i.e., rank values).

In Figure 3.2b, after $\omega = 4$ iterations, all real accounts $\{A, B, C, D, E\}$ collect more trust than fake accounts $\{F, G, H, I\}$. Moreover, the nodes receive the correct ranking of $(D, A, B, C, E, F, G, H, I)$, as sorted by their degree-normalized trust. In particular, all real accounts have higher rank values than fakes, where the smallest difference is $T'_4(E) - T'_4(F) > 40$. Notice that real accounts that are not victims have similar rank values, where the largest difference is $T'_4(D) - T'_4(C) < 12$. These sorted rank values, in fact, could be visualized as a stretched-out step function that has a significant drop near the victim's rank value.

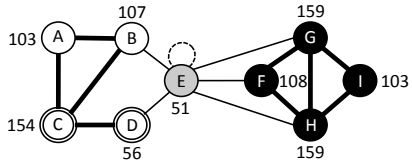
If we allowed the process to converge after $k > 50$ iterations, the fakes collect similar or higher trust than real accounts, following Equation 3.5, as shown in Figure 3.2c. In either case, the attack edges $E_a = \{\{E, G\}, \{E, F\}, \{E, H\}\}$ have a volume of $\text{vol}(E_a) = 0.3$, which is 10 times lower than its value if the graph had unit weights, with $\text{vol}(E_a) = 3$. As we show in Section 3.4, early-termination and propagation rate adjustment are essential for robustness against social infiltration.



(a) Initialization



(b) After 4 iterations



(c) Stationary distribution

Figure 3.2: Trust propagation example

3.3.4 Selecting Trusted Accounts

Íntegro is robust against social infiltration, as it limits the portion of τ that enters G_f by the rate $\text{vol}(E_a)$, regardless of the number of attack edges, $|E_a|$. For the case when there are few attack edges so that G_r and G_f are sparsely connected, $\text{vol}(E_a)$ is already small, even if one keeps $w(v_i, v_j) = 1$ for each attack edge $\{v_i, v_j\} \in E_a$. However, G_r is likely to contain communities [71, 72], where each represents a dense subgraph that is sparsely connected to the rest of the graph. In this case, the propagation of τ in G_r becomes restricted by the sparse inter-community connectivity, especially if V_t is contained exclusively in a single community. We therefore seek a *seed selection strategy* for trusted accounts, which takes into account the existing community structure in the graph.

Seed Selection Strategy

We pick trusted accounts as follows. First, before rate adjustment, we estimate the community structure in the graph using a community detection algorithm called the *Louvain method* [7]. Second, after rate adjustment, we exclude potential victims and pick small samples of nodes from each detected community at random. Third and last, we inspect the sampled nodes in order to verify they correspond to real accounts that are not victims. We initialize the trust only between the accounts that pass manual verification by experts.

In addition to coping with the existing community structure in the graph, this selection strategy is designed to also reduce the negative impact of *seed-targeting* attacks. In these attacks, fake accounts befriend trusted accounts in order to adversely improve their own ranking, as the total trust τ is initially distributed among trusted accounts. By choosing the seeds at random, however, the attacker is forced to guess the seeds among a large number of nodes. Moreover, by choosing multiple seeds, the chance of correctly guessing the seeds is further reduced, while the amount of trust assigned to each seed in lowered. In practice, the number of seeds depends on available resources for manual account verification, with a minimum of one seed per detected community.

Community Detection

We chose the Louvain method because it is both efficient and outputs high-quality partitions. This method iteratively groups closely connected communities together to greedily improve the *modularity* of the partition [93], which is one measure for partition quality. In each iteration, each node represents one community, and well-connected neighbors are greedily combined into the same community. At the end of the iteration, the graph is reconstructed by converting the resulting communities into nodes and adding edges that are weighted by inter-community connectivity. Each iteration takes $O(m)$ time, and only a small number of iterations is required to find the community structure which greedily maximizes the modularity.

While one can apply community detection to identify fake accounts [122], do-

ing so hinges on the assumption that fakes always form tightly-knit communities, which is not necessarily true [134]. This also means fakes can easily evade detection if they establish sparse connectivity among themselves [2]. With Íntegro, we do not make such an assumption. In particular, we consider an attacker *who can befriend a large number of real or fake accounts*, without any formal restrictions.

3.3.5 Computational Cost

For an OSN with n users and m friendships, Íntegro takes $O(n \log n)$ time to complete its computation, end-to-end. We next analyze the running time in detail.

First, recall that users have a limit on how many friends they can have (e.g., 5K in Facebook, 1K in Tuenti), so we have $O(m) = O(n)$. Identifying potential victims takes $O(n \log n)$ time, where it takes $O(n \log n)$ time to train an RF classifier and $O(n)$ time to compute vulnerability scores. Also, weighting the graph takes $O(m)$ time. Detecting communities takes $O(n)$ time, where each iteration of the Louvain method takes $O(m)$ time, and the graph rapidly shrinks in $O(1)$ time. Propagating trust takes $O(n \log n)$ time, as each iteration takes $O(m)$ time and the propagation process iterates for $O(\log n)$ times. Ranking and sorting users by their degree-normalized trust takes $O(n \log n)$ time. So, the running time is $O(n \log n)$.

3.3.6 Security Guarantees

In the following analysis, we consider attackers who establish attack edges with victims uniformly at random. For analytical tractability, we assume the real region is *fast mixing*. This means that it takes $O(\log |V_r|)$ iterations for trust propagation to converge in the real region. We refer the interested reader to Appendix A for a complete analysis, including theoretical background and formal proofs.

Sensitivity to Mixing Time

The ranking quality of Íntegro does not rely on the absolute value of the mixing time in the real region. Instead, it only requires that the whole graph has a longer

mixing time than the real region. Under this condition, the early-terminated propagation process results in a gap between the degree-normalized trust of fakes and real accounts. Ideally, the number of iterations that Íntegro performs is set equal to the mixing time of the real region. Regardless of whether the mixing time of the real region is $O(\log n)$ or larger, setting the number of iterations to this value results in an almost uniform degree-normalized trust among the real accounts [15]. If the mixing time of the real region is larger than $O(\log n)$, the trust that escapes to the fake region is further limited. However, we also run at the risk of starving real accounts that are loosely connected to trusted accounts. This risk is mitigated by placing trusted accounts in many communities, and by dispersing multiple seeds in each community, thereby ensuring that the trust is initiated somewhere close to those real accounts, as described in Section 3.3.4.

Bound on Ranking Quality

The main security guarantee offered by Íntegro is captured by the following theoretical result:

Theorem 1: *Given a social graph with a fast mixing real region and an attacker who randomly establishes attack edges, the number of fake accounts that rank similar to or higher than real accounts after $O(\log n)$ iterations is $O(\mathrm{vol}(E_a)\log n)$.*

Proof sketch. Consider a graph $G = (V, E)$ with a fast mixing real region G_r. As weighting a graph changes its mixing time by a constant factor [102], G_r remains fast mixing after rate adjustment.

After $O(\log n)$ iterations, the trust vector $T_\omega(V)$ does not reach its stationary distribution $T_\infty(V)$. Since trust propagation starts from G_r, the fake region G_f gets only a fraction $f < 1$ of the aggregate trust it should receive in $T_\infty(V)$. On the other hand, as the trust τ is conserved during the propagation process (Equation 3.3), G_r gets $c > 1$ times higher aggregate trust than it should receive in $T_\infty(V)$.

As G_r is fast mixing, each real account $v_i \in V_r$ receives approximately identical rank value of $T'_\omega(v_i) = c \cdot \tau/\mathrm{vol}(V)$, where $\tau/\mathrm{vol}(V)$ is the degree-normalized

trust value in $T_\infty(V)$ (Equations 3.5 and 3.6). Knowing that G_f is controlled by the attacker, each fake $v_j \in V_f$ receives a rank value $T'_\omega(v_j)$ that depends on how the fakes inter-connect to each other. However, since the aggregate trust in G_f is bounded, each fake receives on average a rank value of $T'_\omega(v_j) = f \cdot \tau / \mathrm{vol}(V)$, which is less than that of a real account. In the worst case, an attacker can arrange a set $V_m \subset V_f$ of fake accounts in G_f such that each $v_k \in V_m$ receives a rank value of $T'_\omega(v_k) = c \cdot \tau / \mathrm{vol}(V)$, while the remaining fakes receive a rank value of zero. Such a set cannot have more than $(f/c) \cdot \mathrm{vol}(V_f) = O(\mathrm{vol}(E_a) \log n)$ accounts, as otherwise, f would not be less than 1 and G_f would receive more than it should in $T_\omega(V)$. □

Improvement over SybilRank

Íntegro shares many design traits with SybilRank. In particular, modifying Íntegro by setting $w(v_i, v_j) = 1$ for each $(v_i, v_j) \in E$ will result in a ranking similar to that computed by SybilRank [15]. It is indeed the incorporation of victim classification into user ranking that differentiates Íntegro from other proposals, giving it the unique advantages outlined earlier.

As stated by Theorem 1, the bound on ranking quality relies on $\mathrm{vol}(E_a)$, regardless of how large the set E_a grows. As we weight the graph based on the output of the victim classifier, our bound is sensitive to its classification performance. We next prove that if an OSN operator uses a victim classifier that is *uniformly random*, which means each user account $v_i \in V$ is equally vulnerable with $p(v_i) = 0.5$, then Íntegro is as good as SybilRank in terms of ranking quality:

Corollary 1: *For a uniformly random victim classifier, the number of fakes that rank similar to or higher than real accounts after $O(\log n)$ iterations is $O(|E_a| \log n)$.*

Proof. This classifier assigns each user account $v_i \in V$ a score $p(v_i) = 0.5$. By Equation 3.4, each edge $\{v_i, v_j\} \in E$ is assigned a unit weight $w(v_i, v_j) = 1$, where $\alpha = 0.5$ and $\beta = 2$. By Theorem 1, the number of fake accounts that rank similar to or higher than real accounts after $\omega = O(\log n)$ iterations is $O(\mathrm{vol}(E_a) \log n) = O(|E_a| \log n)$. □

By Corollary 1, Íntegro can outperform SybilRank in its ranking quality by a factor of $O(|E_a|/\mathrm{vol}(E_a))$, given the used victim classifier is better than random. This can be achieved during the cross-validation phase of the victim classifier, which we thoroughly describe and evaluate in what follows.

3.4 Comparative Evaluation

We now present a comprehensive comparative evaluation of Íntegro and SybilRank. First, we explain our choice of SybilRank (Section 3.4.1), after which we outline a summary of the evaluation methodology (Section 3.4.2). This is followed by a detailed description of the used datasets (Section 3.4.3) and system implementation (Section 3.4.4). Next, we evaluate Íntegro for victim classification (Section 3.4.5), and then systematically compare it to SybilRank in terms of ranking quality, sensitivity to seed-targeting attacks, and real-world deployment (Sections 3.4.6–3.4.8). Finally, we evaluate the scalability of Íntegro in terms of its execution time using a synthetic benchmark of large OSNs (Section 3.4.9).

3.4.1 Compared System

We chose SybilRank for two reasons. First, SybilRank uses a similar power iteration method albeit on an unweighted version of the graph. This similarity allowed us to clearly show the impact of leveraging victim classification on fake account detection. Second, SybilRank was shown to outperform known contenders [15], including EigenTrust [60], SybilGuard [138], SybilLimit [139], SybilInfer [19], Mislove's method [122], and GateKeeper [118]. We next contrast these systems to both SybilRank and Íntegro.

SybilGuard [138] and SybilLimit [139] identify fake accounts based on a large number of modified random walks, where the computational cost is $O(\sqrt{mn}\log n)$ in centralized setting like OSNs. SybilInfer [19], on the other hand, uses Bayesian inference techniques to assign each user account a probability score for being fake in $O(n(\log n)^2)$ time per trusted account. The system, however, does not provide

analytical bounds on its ranking quality.

GateKeeper [118], which is a flow-based fake account detection approach, improves over SumUp [117]. GateKeeper relies on strong assumptions that require balanced graphs and costs $O(n \log n)$ time per trusted account or "ticket source."

Viswanath et al. employed Mislove's algorithm [82] to greedily expand a local community around a set of trusted accounts in oder to partition the graph into two communities representing real and fake regions [122]. This algorithm, however, costs $O(n^2)$ time and its detection can be easily evaded if the fakes establish sparse connectivity among themselves [2].

Compared to these systems, SybilRank provides an equivalent or tighter security bound and is more computationally efficient, as it requires $O(n \log n)$ time regardless of number of trusted accounts. Compared to SybilRank, Íntegro provides $O(|E_a|/\mathrm{vol}(E_a))$ improvement on its security bound, requires the same $O(n \log n)$ time, and is robust against social infiltration, unlike all other systems.

3.4.2 Methodology

Our objective is to empirically validate system design from five different aspects: Victim classification, ranking quality, sensitivity to seed-targeting attacks, large-scale deployment, and scalability. To achieve this, we implemented and evaluated both Íntegro and SybilRank using two real-world datasets recently collected from Facebook and Tuenti. We also compared both systems through a production-class deployment at Tuenti in collaboration with its "Site Integrity" team, which has 14 full-time *account analysts* and 10 full-time software engineers who fight spam and other forms of abuse. In order to evaluate system scalability, we tested Íntegro using a benchmark of five synthetic OSNs with an exponentially increasing number of users. We provide more details on our methodology in the following sections.

3.4.3 Datasets

We used two real-world datasets from two different OSNs. The first dataset was collected from Facebook during the study described in Chapter 2, and contained

public user profiles and two graph samples. The second dataset was collected by Tuenti from their production servers, and contained a day's worth of server-cached user profiles. We had to sign a non-disclosure agreement with Tuenti in order to access an anonymized, aggregated version of its user data, with the whole process being mediated by Tuenti's Site Integrity team.

The Ground-truth

For the Facebook dataset, we used the ground-truth of the original study, which we also re-validated for the purpose of this work, as we describe next. For the Tuenti dataset, all accounts were manually inspected and labeled by its account analysts. The inspection included matching user profile photos to its declared age and address, understanding natural language in user posts and messages, examining the user's friends, and analyzing the user's IP address and HTTP-related information.

Facebook

The dataset contained public profile pages of 9,646 real users who received friend requests from fake accounts. Since the dataset was collected in early 2011, we wanted to verify whether these users are still active on Facebook. Accordingly, we revisited their public profiles in June 2013. We found that 7.9% of these accounts were either disabled by Facebook or deactivated by the users. Therefore, we excluded these accounts, ending up with 8,888 accounts, out of which 32.4% were victims who accepted a single friend request sent by a fake posing as a stranger.

The dataset also included two *graph samples* of Facebook, which were collected using a stochastic version of the Breadth-First Search method called "forest fire" [70]. The first graph consisted of 2,926 real accounts with 9,124 friendships (i.e., the real region), 65 fakes with 2,080 friendships (i.e., the fake region), and 748 *timestamped* attack edges. The second graph consisted of 6,136 real accounts with 38,144 friendships, which represented the real region only.

Tuenti

The dataset contained profiles of 60K real users who received friend requests from fake accounts, out of which 50% were victims. The dataset was collected in Feb 10, 2014 from live production servers, where data resided in memory and no expensive, back-end queries were issued. For Tuenti, collecting this dataset was a low-cost and easy process, as it involved reading cached user profiles of a subset of its *daily active users*—users who logged in to Tuenti on that particular day.

3.4.4 Implementation

We developed two implementations for each of Íntegro and SybilRank. In the first implementation, we used the SyPy framework—our open-source Python package for single-machine comparative evaluation of graph-based Sybil accounts detection algorithms. We extended the framework by integrating SciKit-Learn,[2] which is a Python package for machine learning algorithms. We used this implementation for the evaluation presented in Sections 3.4.5–3.4.7. We refer the interested reader to Appendix B for more details on SyPy.

In the second implementation, we used Mahout and Giraph, which are widely-used, open-source distributed machine learning and graph processing systems. We also publicly released the implementation as part of GrafosML, which provides an open-source toolset for big data analytics on top of Mahout and Giraph. We used this implementation for the evaluation presented in Sections 3.4.8–3.4.9.

3.4.5 Victim Classification

We sought to validate the following claim: An OSN can identify potential victims with a probability that is better than random, using strictly low-cost features extracted from readily-available user profiles. We note, however, that it is possible to achieve better classification performance, at the price of a higher computational cost, by using advanced learning algorithms with temporal activity features [47].

[2] http://scikit-learn.org

Features

As described in Table 3.1, we extracted features from both datasets to generate feature vectors. The only requirement for feature selection was to have each feature value available for all users in the dataset, so that the resulting feature vectors are complete. For the Facebook dataset, we were able to extract 18 features from public user profiles. For Tuenti, however, the dataset was limited to 14 features, but contained user features that are not publicly accessible.

In Table 3.1, the RI score stands for the *relative importance* of the feature, which we define in the upcoming discussion. An RI score of "N/A" means the feature was not available for the corresponding dataset. A *k-categorical* feature means the feature can have one value out of k unique categories. For example, boolean features are 2-categorical.

Classifier Tuning

The RF learning algorithm is an *ensemble* algorithm, where a set of decision trees are constructed at training time. When evaluating the classifier on new data (i.e., unlabeled feature vectors), the decisions from all trees are combined using a majority voting aggregator [12]. Each decision tree in the forest uses a small random subset of available features in order to decrease the *generalization error*, which measures how well the trained classifier generalizes to unseen data [47]. As shown in Figure 3.3, we performed parameter tuning to calibrate the RF classifier. In particular, we used the out-of-bag error estimates computed by the RF algorithm to numerically find the best number of decision trees and the number of features for each tree, so that the prediction variance and bias are controlled across the trees. For the Facebook dataset, we used 450 decision trees, where each tree had 3 features picked at random out of 18 features. For the Tuenti dataset, we used 500 decision trees, where each tree had 7 features picked at random out of 14 features.

Feature	Brief description	Type	RI Score (%)	
			Facebook	Tuenti
User activity:				
Friends	Number of friends the user had	Numeric	100.0	84.5
Photos	Number of photos the user shared	Numeric	93.7	57.4
Feed	Number of news feed items the user had	Numeric	70.6	60.8
Groups	Number of groups the user was member of	Numeric	41.8	N/A
Likes	Number of likes the users made	Numeric	30.6	N/A
Games	Number of games the user played	Numeric	20.1	N/A
Movies	Number of movies the user watched	Numeric	16.2	N/A
Music	Number of albums or songs the user listened to	Numeric	15.5	N/A
TV	Number of TV shows the user watched	Numeric	14.2	N/A
Books	Number of books the user read	Numeric	7.5	N/A
Personal messaging:				
Sent	Number of messages sent by the user	Numeric	N/A	53.3
Inbox	Number of messages in the user's inbox	Numeric	N/A	52.9
Privacy	Privacy level for receiving messages	5-categorical	N/A	9.6
Blocking actions:				
Users	Number of users blocked by the user	Numeric	N/A	23.9
Graphics	Number of graphics (photos) blocked by the user	Numeric	N/A	19.7
Account information:				
Last updated	Number of days since the user updated the profile	Numeric	90.77	32.5
Highlights	Number of years highlighted in the user's time-line	Numeric	36.3	N/A
Membership	Number of days since the user joined the OSN	Numeric	31.7	100
Gender	User is male or female	2-categorical	13.8	7.9
Cover picture	User has a cover picture	2-categorical	10.5	< 0.1
Profile picture	User has a profile picture	2-categorical	4.3	< 0.1
Pre-highlights	Number of years highlighted before 2004	Numeric	3.9	N/A
Platform	User disabled third-party API integration	2-categorical	1.6	< 0.1

Table 3.1: Low-cost features extracted from Facebook and Tuenti

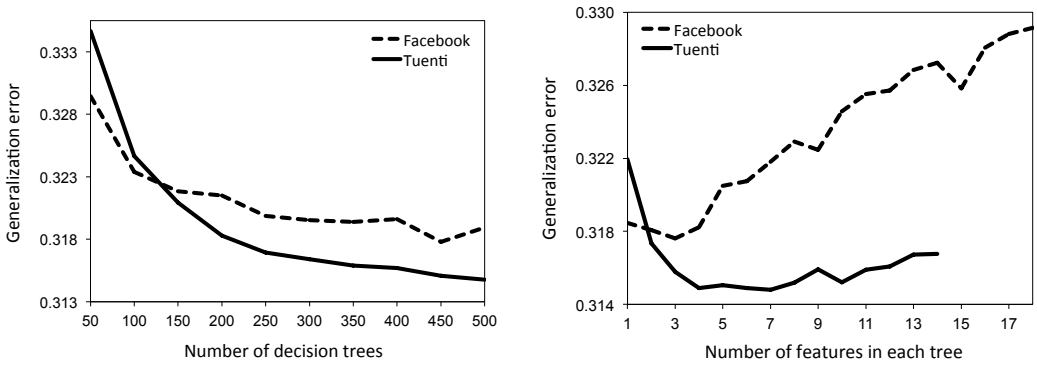


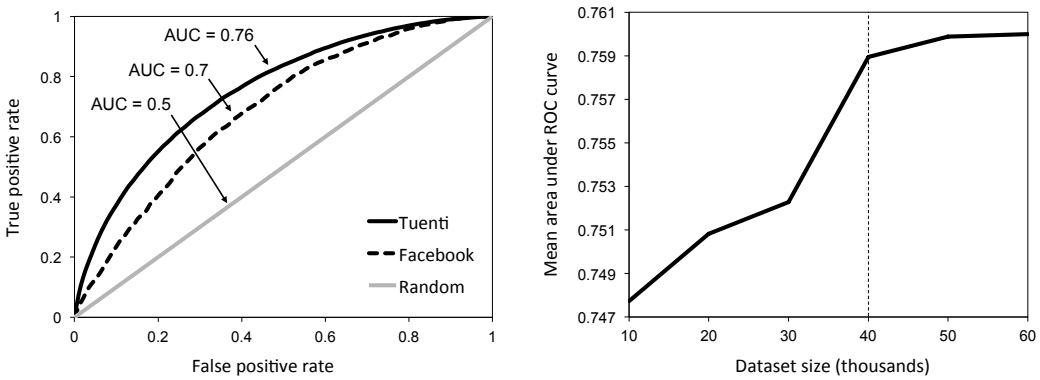
Figure 3.3: Victim classifier tuning

Validation Method

To evaluate the accuracy of the victim classifier, we performed a 10-fold, *stratified cross-validation* method [47] using the RF learning algorithm after parameter tuning. First, we randomly partitioned the dataset into 10 equally-sized sets, with each set having the same percentage of victims as the complete dataset. We next trained an RF classifier using 9 sets and tested it using the remaining set. We repeated this procedure 10 times (i.e., folds), with each of the sets being used once for testing. Finally, we combined the results of the folds by computing the mean of their true-positive rate (TPR) and false-positive rate (FPR).

Performance Metrics

The output of the classifier depends on its *operating threshold*, which is a cutoff value in the prediction probability after which the classifier identifies a user as a potential victim. In order to capture the trade-off between TPR and FPR in single curve, we repeated the cross-validation method under different threshold values using a procedure known as *receiver operating characteristics* (ROC) analysis. In ROC analysis, the closer the curve is to the top-left corner at point $(0, 1)$ the better the classification performance is. The quality of the classifier can be quantified with a single value by calculating the *area under its ROC curve* (AUC) [47], so



(a) ROC Analysis **(b)** Sensitivity to dataset size

Figure 3.4: Victim classification using the RF algorithm

that an AUC of 1 means a perfect classifier, while an AUC of 0.5 means a random classifier. The victim classifier has to be better than random with AUC > 0.5.

We also recorded the *relative importance* (RI) of features used for the classification. The RI score is computed by the RF learning algorithm, and it describes the relative contribution of each feature to the predictability of the label—being a victim or a non-victim—when compared to all other features [12].

Results

For both datasets, the victim classifier ended up with an AUC greater than 0.5, as depicted in Figure 3.4a. In particular, for the Facebook dataset, the classifier delivered an AUC of 0.7, which is 40% better than random. For the Tuenti dataset, on the other hand, the classifier delivered an AUC of 0.76, which is 52% better than random. Also, increasing the dataset size to more than 40K feature vectors did not significantly improve the AUC in cross-validation, as show in Figure 3.4b. This means an OSN operator can train a victim classifier using a relatively small dataset, so fewer accounts need to be manually verified.

We also experimented with another two widely-used learning algorithms for victim classification, namely, Naïve Bayes (NB) and SVM [47]. However, both of these algorithms resulted in smaller AUCs on both datasets. In particular, for

the Facebook dataset, the NB classifier achieved an AUC of 0.63 and the SVM classifier achieved an AUC of 0.57. Similarly, for the Tuenti dataset, the NB classifier achieved an AUC of 0.64 and the SVM classifier achieved an AUC of 0.59. This result is not surprising as ensemble learning algorithms, such as the RF algorithm, achieve better predictive performance in case individual classifiers are "weak," meaning they have small AUCs but are still better than random [47].

3.4.6 Ranking Quality

We compared Íntegro against SybilRank in terms of their ranking quality under various attack scenarios, where ideally real accounts should be ranked higher than fake accounts. Our results are based on the average of at least 10 runs, with error bars reporting 95% confidence intervals (CI), when applicable. For this comparison, we picked the Facebook dataset because it included both feature vectors and graph samples.

Infiltration Scenarios

We considered two real-world attack scenarios which have been shown to be successful in practice. In the first scenario, attackers establish attack edges by targeting users with whom their fakes have mutual friends [10]. Accordingly, we used the first Facebook graph which contained timestamped attack edges, allowing us to replay the infiltration by 65 socialbots ($n = 2,991$ and $m = 11,952$). We refer to this scenario as the *targeted-victim* attack.

In the second scenario, we attackers establish attack edges by targeting users at random [15]. We designated the second Facebook graph as the real region. We then generated a synthetic fake region consisting of 3,068 fakes with 36,816 friendships using the small-world graph model [130]. We then added 35,306 attack edges at random between the two regions ($n = 9,204$ and $m = 110,266$). As suggested in related work [137], we used a relatively large number of fakes and attack edges in order to stress-test both systems under evaluation. We refer to the this scenario as the *random-victim* attack.

Edge Weights

For each infiltration scenario, we deployed the previously trained victim classifier in order to assign new edge weights. As we injected fakes in the second scenario, we generated their feature vectors by sampling each feature distribution of fakes from the first scenario.[3] We also assigned edge weights using a victim classifier that simulates two operational modes. In the first mode, the classifier outputs the *best* possible victim predictions with an AUC≈1 and probabilities greater than 0.95. In the second mode, the classifier outputs uniformly *random* predictions with an AUC≈0.5. We used this multi-mode classifier to evaluate the theoretical best and practical worst case performance of Íntegro (see the legend in Figure 3.5).

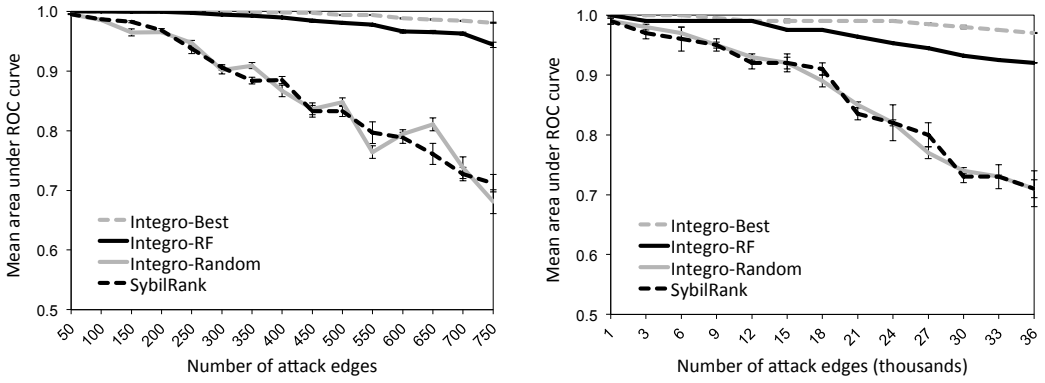
Evaluation Method

To evaluate each system's ranking quality, we ran the system using both infiltration scenarios starting with a single attack edge. We then added another attack edge, according to its timestamp if available, and repeated the experiment. We kept performing this process until there were no more edges to add. At the end of each run, we measured the resulting AUC of each system, as explained next.

Performance Metric

For the resulting ranked list of accounts, we performed ROC analysis by moving a pivot point along the list, starting from the bottom. If an account is behind the pivot, we marked it as fake; otherwise, we marked it as real. Given the ground-truth, we measured the TPR and the FPR across the whole list. Finally, we computed the corresponding AUC, which in this case *quantifies the probability that a random real account is ranked higher than a random fake account*.

[3] We excluded the "friends" feature, as it can be computed from the graph.



(a) Targeted-victim attack **(b)** Random-victim attack

Figure 3.5: Ranking quality under each infiltration scenario (CI=95%)

Seeds and Iterations

In order to make the chance of guessing seeds very small, we picked 100 trusted accounts that are non-victim, real accounts. We used a total trust that is equal to n, the number of nodes in the given graph. We also performed $\lceil \log_2(n) \rceil$ iterations for both Íntegro and SybilRank.

Results

Íntegro consistently outperformed SybilRank in ranking quality, especially as the number of attack edges increased. Using the RF classifier, Íntegro resulted in an AUC which is always greater than 0.92, which is up to 30% improvement over SybilRank in each attack scenario, as shown in Figure 3.5.

In each infiltration scenario, both systems performed well when the number of attack edges was relatively small. In other words, the fakes were sparsely connected to real accounts and so the regions were easily separated. As SybilRank limits its number of fakes that can outrank real accounts by the number of attack edges, its AUC degraded significantly as more attack edges were added to each graph, all the way down to 0.71. Íntegro, however, maintained its performance, with at most 0.07 decrease in AUC, even when the number of attack edges was relatively

large. Notice that Íntegro performed nearly as good as SybilRank when a random victim classifier was used, but performed much better when the RF classifier was used instead. This clearly shows the impact of leveraging victim classification on fake account detection.

3.4.7 Sensitivity to Seed-targeting Attacks

Sophisticated attackers might obtain a full or partial knowledge of which accounts are trusted by the OSN operator. As the total trust is initially distributed among these accounts, an attacker can adversely improve the ranking of the fakes by establishing attack edges directly with them. We next evaluate both systems under two variants of this seed-targeting attack.

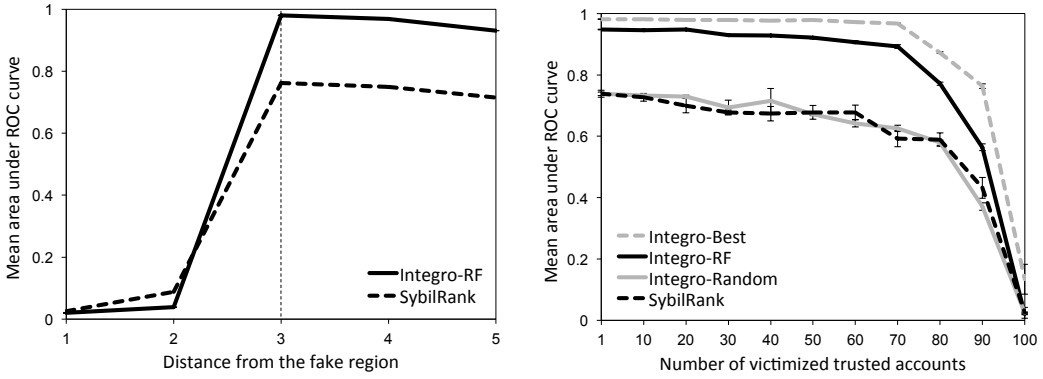
Attack Scenarios

We focus on two main attack scenarios. In the first scenario, the attacker targets accounts that are *k nodes away from all trusted accounts*. This means that the length of the shortest path from any fake account to any trusted account is exactly $k+1$, representing the *distance* between the seeds and the fake region. For $k = 0$, each trusted account is a victim and located at a distance of 1. We refer to this scenario, which assumes a resourceful attacker, as the *distant-seed* attack.

In the second scenario, attackers have only a partial knowledge and target k trusted accounts at random. We refer to this scenario as the *random-seed* attack.

Evaluation Method

To evaluate the sensitivity of each system to a seed-targeting attack, we used the first Facebook graph to simulate each attack scenario. We implemented this by replacing the endpoint of each attack edge in the real region with a real account picked at random from a set of candidates. For the first scenario, a candidate account is one that is k nodes away from all trusted accounts. For the second scenario, a candidate account is simply any trusted account. We ran experiments for both systems using different values of k and measured the corresponding AUC



(a) Distant-seed attack **(b)** Random-seed attack

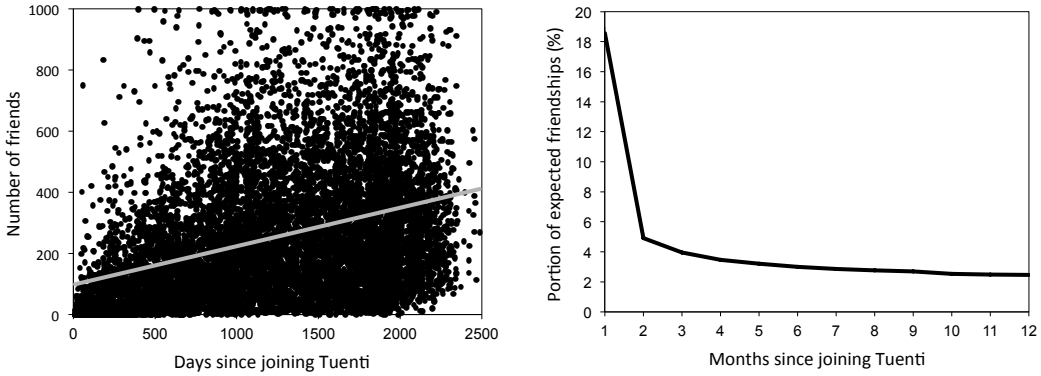
Figure 3.6: Ranking sensitivity to seed-targeting attacks (CI=95%)

at the end of each run.

Results

In the first attack scenario, both systems had a poor ranking quality when the distance was small, as illustrated in Figure 3.6a. Because Íntegro assigns low weights to edges incident to victim accounts, the trust that escapes to the fake region is less likely to come back into the real region. This explains why SybilRank had a slightly better AUC for distances less than 3. However, once the distance was larger, Íntegro outperformed SybilRank ,as expected from earlier results.

In the second attack scenario, the ranking quality of both systems degraded, as the number of victimized trusted accounts increased, where Íntegro consistently outperformed SybilRank, as illustrated in Figure 3.6b. Notice that by selecting a larger number of trusted accounts, it becomes much harder for an attacker to guess which account is trusted, while the gained benefit per victimized trusted account is further reduced.



(a) Users connectivity **(b)** Friendship growth over time

Figure 3.7: Preprocessing for system deployment

3.4.8 Deployment at Tuenti

We deployed both systems on a snapshot of Tuenti's daily active users graph in February 6, 2014. The graph consisted of several million nodes and tens of millions of edges. We had to mask out the exact numbers due to a non-disclosure agreement with Tuenti. After initial analysis of the graph, we found that 96.6% of nodes and 94.2% of edges belonged to one *giant connected component* (GCC). Therefore, we focused our evaluation on this GCC.

Preprocessing

Using a uniform random sample of 10K users, we found that new users have weak connectivity to others due to the short time they have been on Tuenti, as shown in Figure 3.7a. In particular, there was a positive correlation between number of days since a user joined Tuenti and how well-connected the user is in terms of number of friends (Pearson's $r = 0.36$). In fact, *93% of all new users who joined Tuenti in the last 30 days had weak connectivity of 46 friends or less*, much smaller than the average of 254 friends. If these users were included in our evaluation, they would end up receiving low ranks, which would lead to false positives.

To overcome this hurdle, we estimated the period after which users accumulate at least 10% of the average number of friends in Tuenti. To achieve this, we used

a uniformly random sample of 10K real users who joined Tuenti over the last 77 months. We divided the users in the sample into buckets representing how long they have been active members. We then calculated the average number of new friendships they made after every other month. As illustrated in Figure 3.7b, users accumulated 53% of their friendships during the first 12 months. In addition, 18.6% of friendships were made after one month since joining the network. To this end, we decided to defer the consideration of users who have joined in the last 30 days since Feb 6, 2014, which represented only 1.3% of users in the GCC.

Community Detection

We applied the Louvain method on the preprocessed GCC. The method finished quickly after just 5 iterations with a high modularity score of 0.83, where a value of 1 corresponds to a perfect partitioning. In total, we found 42 communities and the largest one consisted of 220,846 nodes. In addition, 15 communities were relatively large containing more than 50K nodes. Tuenti's account analysts verified 0.05% of the nodes in each detected community, and designated these nodes as trusted accounts for both systems.

Performance Metric

As the number of users in the processed GCC is large, it was infeasible to manually inspect and label each account. This means that we were unable to evaluate the system using ROC analysis. Instead, we attempted to determine the percentage of fake accounts at equally-sized intervals in the ranked list. We accomplished this in collaboration with Tuenti's analysts by manually inspecting a user sample in each interval in the list. This percentage is directly related to the *precision* of fake account detection, which is a performance metric typically used to measure the ratio of relevant items over the top-k highest ranked items in terms of relevance [51].

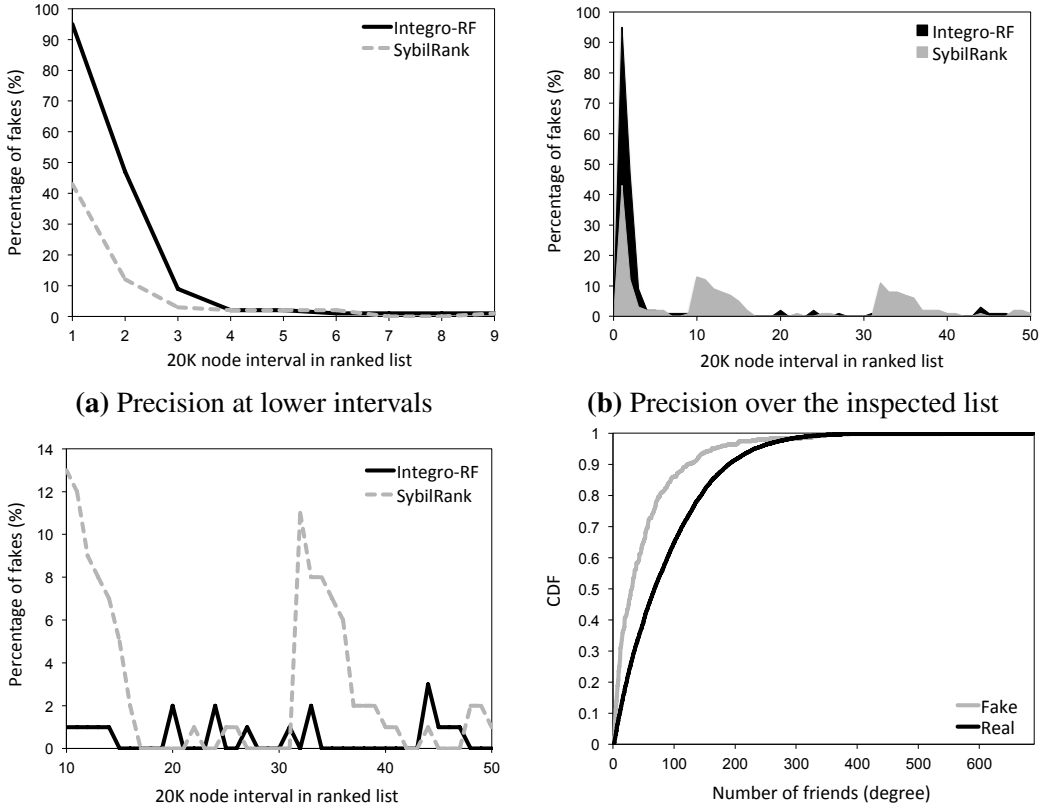
Evaluation Method

We utilized the previously trained victim classifier in order to weight a copy of the graph. We then ran both systems on two versions of the graph (i.e., weighted and unweighted) for $\lceil \log_2(n) \rceil$ iterations, where n is number of accounts in the graph. After that, we examined the ranked list of each system by inspecting the first lowest-ranked one million user accounts. We decided not to include the complete range due to confidentiality reasons, because otherwise one could precisely estimate the actual number of fakes in Tuenti. We randomly selected 100 users out of each 20K user interval for inspection in order to measure the percentage of fakes in the interval, that is, the precision.

Results

As illustrated in Figure 3.8a, Íntegro resulted in 95% precision in the lowest 20K ranking user accounts, as opposed to 43% by SybilRank and 5% by Tuenti's user-based abuse reporting system. The precision dropped dramatically as we went up in the list, which means our ranking scheme placed most of the fakes at the bottom of the ranked list, as shown in Figure 3.8b.

Let us consider SybilRank's ranking, as shown in Figures 3.8a and 3.8c. The precision, starting with 43% for the first interval, gradually decreased until rising again at the 10th interval. This pattern repeated at the 32nd interval as well. We inspected the fake accounts at these intervals and found that they belonged to three different, large communities. In addition, these fakes had a large number of friends, much larger than the average of 254 friends. In particular, the fakes from the 32nd interval onwards had more than 300 friends, with a maximum of up to 539. Figure 3.8d shows the degree distribution for both *verified* fake and real accounts. This figure suggests that fakes tend to create many attack edges with real accounts, which confirms earlier findings on other OSNs such as Facebook [10]. Also, this behavior explains why Íntegro outperformed SybilRank in user ranking quality; these high degree fakes received lower ranks as most of their victims were identified by the classifier.



(a) Precision at lower intervals

(b) Precision over the inspected list

(c) Precision at higher intervals

(d) Node degree distribution

Figure 3.8: Deployment results at Tuenti

SybilRank in Retrospect

SybilRank was initially evaluated on Tuenti, where it effectively detected a significant percentage of the fakes [15]. The original evaluation, however, pruned excessive edges of nodes that had a degree greater than 800, which include a non-disclosed number of fakes that highly infiltrated Tuenti. Also, the original evaluation was performed on the whole graph, which included many dormant accounts. In contrast, our evaluation was based on the daily active users graph in order to focus on active fake accounts that could be harmful. While this change limited the number of fakes that existed in the graph, it has evidently revealed the ineffectiveness of SybilRank under social infiltration. Additionally, the original

evaluation showed that 10–20% of fakes received high ranks, a result we also attest, due to the fact that these fake accounts had established many attack edges. On the other hand, Íntegro has 0–2% of fakes at these high intervals, and so it delivers an order of magnitude better precision than SybilRank.

3.4.9 Scalability

We next evaluate the scalability of Íntegro in terms of its execution time as the number of users in the OSN increase. Accordingly, we used the distributed version of our software implementation (Section 3.4.4), and deployed it on a commodity computer cluster for benchmarking, as follows.

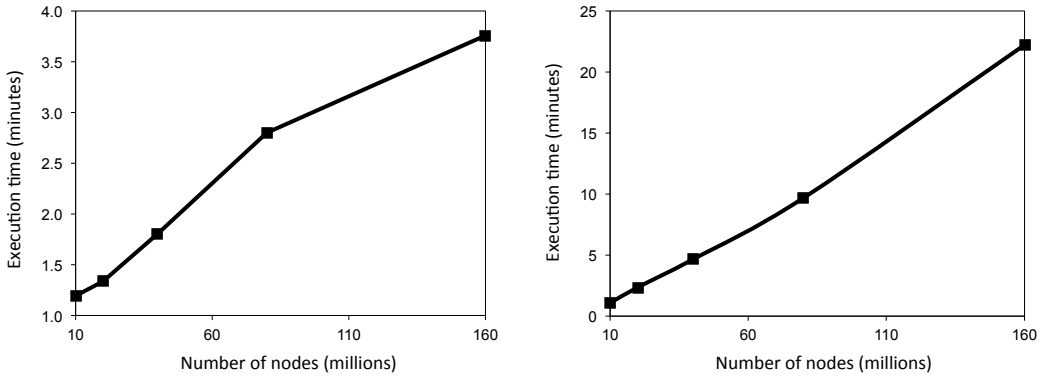
Benchmark

We deployed Íntegro on an Amazon Elastic MapReduce[4] cluster. The cluster consisted of one *m1.small* instance serving as a master node and 32 *m2.4xlarge* instances serving as slave nodes. We employed the small-world graph model [130] to generate 5 graphs with an exponentially increasing number of nodes. For each one of these graphs, we used the Facebook dataset to randomly generate all feature vectors with the same distribution for each feature. We then ran Íntegro on each of the generated graphs and measured its execution time.

Results

Íntegro achieved a nearly linear scalability with the number of nodes in a graph, as illustrated in Figure 3.9. Excluding the time required to load the 160M node graph into memory, which is about 20 minutes for a non-optimized data format, it takes less than 2 minutes to train an RF victim classifier and compute vulnerability scores for nodes on Mahout. It also takes less than 25 minutes to weight the graph, rank nodes, and finally sort them on Giraph. This makes Íntegro computationally practical even for large OSNs such as Facebook.

[4] http://aws.amazon.com/elasticmapreduce



(a) Victim classification on Mahout **(b)** User ranking on Giraph

Figure 3.9: System scalability on distributed computing platforms

3.5 Discussion

As presented in Section 3.3.6, Íntegro's security guarantee is sensitive to the performance of the deployed victim classifier, which is formally captured by $\text{vol}(E_a)$ in the bound $O(\text{vol}(E_a)\log n)$, and can be practically measured by its AUC.

3.5.1 Robustness of User Ranking

As illustrated in Figure 3.5, improving the AUC of the victim classifier from random with AUC ≈ 0.5, to actual with AUC$= 0.7$, and finally to best with AUC ≈ 1 consistently improved the resulting ranking in terms of its AUC. Therefore, a higher AUC in victim classification leads to a higher AUC in user ranking. This is the case because the ROC curve of a victim classifier monotonically increases, so a higher AUC implies a higher true positive rate (TPR). In turn, a higher TPR means more victims are correctly identified, and so more attack edges are assigned lower weights, which evidently leads to a higher AUC in user ranking.

Regardless of the used victim classifier, the ranking quality decreases as the number of attack edges increases, as shown in Figure 3.5. This is the case because even a small false negative rate (FNR) in victim classification means more attack edges, which are indecent to misclassified victims, are assigned high weights,

leading to a lower AUC in user ranking.

3.5.2 Maintenance and Impact

While an attacker does not control real accounts nor their activities, it can still trick users into befriending fakes. In order to achieve a high-quality ranking, the victim classifier should be regularly retrained to capture new and changing user behavior in terms of susceptibility to social infiltration. This is, in fact, the case for supervised machine learning when applied to computer security problems [104]. Also, as the ranking scheme is sensitive to seed-targeting attacks, the set of trusted accounts should be regularly updated and validated in order to reduce the negative impact of these attacks, even if they are unlikely to succeed (Section 3.3.4).

By using Íntegro, Tuenti requires nearly 67 man hours to manually validate the 20K lowest ranking user accounts, and discover about 19K fake accounts instead of 8.6K fakes with SybilRank. With its user-based abuse reporting system that has 5% hit rate, and assuming all fakes get reported, Tuenti would need 1,267 man hours instead to discover 19K fake accounts (18.9 times more man hours). As such, this improvement has been useful to both Tuenti and its users.

3.5.3 Limitations

Íntegro is not a stand-alone fake account detection system. It is intended to complement existing detection systems and is designed to detect automated fake accounts that befriend many victims for subsequent attacks. In what follows, we outline two limitations which are inherited from SybilRank [15] and similar random walk-based ranking schemes [137].

Design is Limited to only Undirected Social Graphs

In other words, OSNs whose users declare lateral relationships are not expected to benefit from our proposal. This is the case because directed graphs, in general, have a significantly smaller mixing time than their undirected counterparts [84],

which means a random walk on such graphs will converge in a much small number of steps, rendering short random walks unsuitable for robust user ranking.

Deployment Delays the Consideration of New User Accounts

This means that an OSN operator might miss the chance to detect fakes at their early life-cycle. However, as shown in Figure 3.7a, only 7% of new users who joined Tuenti in the last month had more than 46 friends. To estimate the number of fakes in new accounts, we picked 100 accounts at random for manual verification. We found that only 6% of these accounts were fake, and the most successful fake account had 103 victims. In practice, the decision of whether to exclude these account is operational, and it depends on the actions taken on low-ranking users. For example, an operator can enforce abuse mitigation technique, as discussed in Section 3.1.3, against low-ranking users, where false positives can negatively affect user experience but slow down fake accounts that just joined the network. This is a security/usability trade-off which we leave to the operator to manage. Alternatively, the operator can use fake account detection systems that are designed to admit legitimate new users using, for example, a vouching process [131].

3.6 Summary

Detecting fake accounts protects both OSN operators and their users from various malicious activities. Most detection mechanisms attempt to classify user accounts as real or fake by analyzing either user-level activities or graph-level structures. These mechanisms, however, are not robust against adversarial attacks in which fake accounts cloak their operation with patterns resembling real user behavior.

This work stemmed Findings 2 and 4 from Chapter 2, and demonstrated that victims—real accounts whose users have accepted friend requests sent by fakes—form a distinct classification category that is useful for designing robust detection mechanisms. To start with, as attackers have no control over victim accounts and cannot alter their activities, a victim account classifier which relies on user-level activities to identify potential victims is relatively hard to circumvent. Moreover,

as fakes are directly connected to victims, a graph-based fake account detection mechanism that leverages victim classification is robust against adverse manipulations of the graph, social infiltration in particular.

To validate this new approach, we designed and evaluated Íntegro—a robust and scalable defense system that leverages victim classification to rank most real accounts higher than fakes, so that OSN operators can take actions against low-ranking fake accounts. In particular, Íntegro starts by identifying potential victims from user-level activities using supervised machine learning. After that, it annotates the graph by assigning lower weights to edges incident to potential victims than others. Finally, Íntegro ranks user accounts based on the landing probability of a short random walk that starts from a known real account. As this walk is unlikely to traverse low-weight edges in few steps and land on fakes, Íntegro achieves the desired ranking.

We implemented Íntegro using widely-used, open-source distributed computing platforms, where it scaled nearly linearly. We also evaluated Íntegro against SybilRank, which is the state-of-the-art in fake account detection, using real-world datasets and a large-scale deployment at Tuenti—the largest OSN in Spain. We showed that Íntegro significantly outperforms SybilRank in user ranking quality, with the only requirement that the employed victim classifier is better than random. Moreover, the deployment of Íntegro at Tuenti resulted in up to an order of magnitude higher precision in fake account detection, as compared to SybilRank.

At the moment, Tuenti uses Íntegro in production in order to thwart fake accounts in the wild, with at least 10 time better precision than their older detection method and hundreds of hours less time spent for manual account verification.

Chapter 4

Discussion and Research Directions

We now discuss different directions in defending against automated fake accounts in OSNs. We first elaborate that *preventing* social infiltration boils down to solving a set of hard socio-technical challenges (Section 4.1). After that, we explore a new defense direction which involves "risky" account admission control (Section 4.2). Finally, we advocate that leveraging victim prediction in OSNs is a new security paradigm which can benefit different defense mechanisms (Section 4.3).

4.1 Challenges for Preventive Countermeasures

When operated as part of a socialbot network (SbN), an OSN can defend against large-scale social infiltration by following at least one of three defense strategies: (1) preventing its operation in the first place, (2) detecting its operation as early as possible, or (3) limiting the advantage an attacker gains from its operation.

Preventing SbN operation can be achieved by eliminating its enabling factors, that is, by fixing at least one of the vulnerabilities outlined in Chapter 2. In what follows, we demonstrate that doing so leads to a set of hard socio-technical challenges that relate to web automation, online-offline identity binding, and usable security. This means that *an OSN has more leverage in detecting and limiting the abusive operation of an SbN rather than preventing its operation in the first place*.

4.1.1 Web Automation

In order to simulate a user browsing an OSN platform (e.g., Facebook's desktop website), the attacker can employ *web automation* techniques, which include methods for solving CAPTCHAs, creating and populating multiple OSN accounts, crawling the social graph, and executing online social activities. Preventing this automation, however, requires solving at least one of the following challenges.

Challenge 1: *Design a reverse Turing test that is usable and effective even against "illegitimate" human solvers.*

A *reverse Turing test*, such as CAPTCHA [124], is a challenge-response test which is administered by a machine and is designed to tell humans and machines apart. A *perfect* Turing test presents a problem that is easy enough for all humans to solve, but is still impossible for a machine or an automation software to pass. Unfortunately, even a perfect test is ineffective if humans are exploited to solve the test in an *illegitimate setting*: The situation where human users are employed or tricked into solving reverse Turing tests that are not addressed to them. Under this setting, we refer to such a human solvers as *illegitimate*.

Eliminating the economic incentives for underground businesses that employ illegitimate human solvers is a first step towards tackling this challenge [87], but it does not solve it as legitimate users can be tricked and situated into illegitimate settings, which is the case for the Koobface botnet [112]. This demands the design of new reverse Turing tests that are resilient to even those illegitimate users, which we believe is generally difficult to achieve.

Fast-response CAPTCHAs, for example, require the test to be solved in a relatively shorter time, as opposed to typical implementations. This makes it more difficult for automation scripts to pass the test, as they require extra time to relay the test, solve it and respond back. Fast-response CAPTCHAs, however, are expected to put more pressure on legitimate users who require easy and fast access to online services, and could potentially repel them away from using them.

Alternatively, authenticating users via their social knowledge (e.g., whether they can identify their friends from photos), can be used as an effective test that

is challenging for illegitimate users to solve [136]. Other that its usability issues, Kim et al. [65] show that it is relatively easy to circumvent such a social authentication scheme by either guessing, automated face recognition, or social engineering.

Challenge 2: *Effectively limit large-scale Sybil crawls of OSNs without restricting users' social experience.*

A *large-scale crawl* is a malicious activity where an attacker manages to crawl large portions of a target OSN, including both the social graph and all accessible user attributes. Today, large-scale crawls are mitigated by employing a network-wide audit service, which limits the number of profiles a user can view per account or IP address in a given period of time [104]. This, however, can be circumvented by creating a set of fake accounts and then performing *Sybil crawling* on a large scale, typically using a botnet with multiple IP addresses [112].

To overcome this drawback, one can use the knowledge about the social graph to effectively limit Sybil crawling. Genie [85], for example, is a system that models the trust between users in an OSN as a *credit network*, where a user can view the profile of another user only if the path between them in the social graph has enough credits to satisfy the operation. If an attacker who controls many fake accounts attempts to crawl the OSN on a large scale, then Genie guarantees that the attacker will exhaust all the credits on the paths connecting the fakes to the rest of the network, thus limiting large-scale Sybil crawling. This approach, however, is based on the assumption that it would be hard for an attacker to establish a large number of social relationships with other users, which is not case as we showed in Section 2.4.5.

Challenge 3: *Detect abusive and automated usage of OSN platforms and social APIs across the Web.*

In concept, *malicious automation* represents the situation in which an attacker scripts his way of consuming system's resources in order to cause damage or harm to the system itself or its users. *Abusive automation*, on the other hand, is less

severe because the attacker exploits the offered service in violation of the declared terms of service (ToS). From OSN operator's standpoint, all HTTP requests come from either a browser or through the social API, which is intentionally provided to support automation. Requests that are not associated with a browsing session, that is, those that do not append the required session cookies, can be easily detected and dealt with. With web automation, however, an attacker can simulate an OSN user and make all requests look as if they originate from a browser. Moreover, the patterns at which these requests are made can be engineered in such a way that makes them fall under the normal traffic category. In order to uncover adversarial campaigns, it is important to reliably identify whether such requests come from a human or a bot, along with means to distinguish patterns of abusive activities, even if the attacker has a knowledge of the used classification techniques.

Looking for regularities in the times at which requests are made, for example, can be used to detect automation in OSNs [140]. This, however, can be easily circumvented by simply mimicking the times and irregularities at which a human user makes such requests.

4.1.2 Identity Binding

Most of the challenges we presented so far are difficult due to the capability of the attacker to mount the Sybil attack. This leads us to the following challenge:

Challenge 4: *Guarantee an anonymous, yet credible, online-offline identity binding in open-access systems.*

Douceur [22] showed that without a centralized trusted party that certifies online identities, Sybil attacks are always possible except under extreme and unrealistic assumptions of resource parity and coordination among participating entities. Thus, limiting the number of fake accounts by forcing a clean mapping between online and offline identities is widely recognized as a hard problem, especially given the scalability requirements of today's open-access OSNs.

Arguably, one way to tackle this challenge is to rely on governments for online identity management, just as in offline settings. The *open government* ini-

tiative [111], for example, enables U.S. citizens to easily and safely engage with U.S. government websites using open identity technologies such as OpenID. This, however, requires creating *open trust frameworks* [111] that enable these websites to accept identity credentials from third-party identity providers, a task that involves solving challenging issues related to identity anonymity, scalability, security, technology incentives and adoption [75, 110].

4.1.3 Usable Security

As part of computer security, *usable security* aims to provide the users with security controls they can understand and privacy they can control [18]. In OSNs such as Facebook, there appears to be a growing gap between what the user expects from a privacy control and what this control does in reality [73]. Even if the most sophisticated OSN security defense is in place, an OSN is still vulnerable to many threats, such as social phishing [58], in case its users find it puzzling to make basic online security or privacy decisions. This gives us strong motives to study the human aspect of the OSN security chain, which is by itself a challenge.

Challenge 5: *Develop OSN security and privacy controls that help users make informed decisions.*

Designing security controls that better communicate the risks of befriending a stranger, for example, might be effective against automated social engineering. This, however, requires eliciting and analyzing the befriending behavior of users, including the factors that influence their befriending decisions, in order to inform a user-centered design for such controls.

Rashtian et al. [97] conducted qualitative and quantitative studies to explain how various factors influence users when accepting friend requests in Facebook. They found that few factors significantly impact the user's decision, namely, knowing the requester in real world, having common hobbies or interests, having mutual friends, and the closeness of mutual friends. The study, however, does not evaluate how an OSN operator can utilize these factors to improve user controls in order to limit the success of social infiltration in OSNs.

4.2 Account Admission Control in OSNs

As presented in Chapter 3, fake account detection represents a *reactive* defense strategy, in which new accounts are admitted to the OSN and then classified. On the other hand, admission control represents a *proactive* defense strategy, in which accounts are provisioned and controlled based on user or OSN defined criteria, all before being admitted to the OSN or given full access to its available services. Although admission control can lead to more resilient defense mechanisms, it is based on the presumption that "users are guilty until proven innocent," which introduces serious concerns related to user experience and the growth of the OSN in terms of its user-base [104]. In what follows, we review prominent mechanisms for admission control in OSN, which can be deployed along side other fake account detection mechanisms, including Íntegro.

4.2.1 Vouching

Xie et al. proposed to identify newly registered, real accounts as early as possible in order to grant them full access to available services [131]. The authors developed a vouching process that is resilient to attackers, where existing real accounts vouch for new accounts. By carefully monitoring vouching via social community structures, they were able to admit 85% of real accounts while reducing the percentage of admitted fake accounts from 44% to 2.4%, using a dataset provided by Hotmail and another one collected from Twitter.

4.2.2 Service Provisioning

Mislove et al. were among the first to dynamically limit available OSN services to unknown fake accounts by modeling lateral trust relationships between users as a credit network [81]. The authors developed a technique that assigns credit values to friendships such that an account is able to send a friend request, for example, only if there is a path with available credit from the sender to the receiver. Mondal et al. utilized this approach to limit large-scale crawls in OSNs [85].

4.2.3 User Education and Security Advice

Kim et al. proposed to visualize the trust between users in order to help them better authenticate those who request their friendship in OSNs such as Facebook [66]. Based on prior social science research demonstrating that the *social tie strength* is a strong indicator of trust, the authors developed a tool to visualize the social tie strength between the receiver and the sender of a friend request, based on features of their mutual friends such as their interaction frequency, communication reciprocity, recency, and length. To validate their approach, the authors conducted a survey with 93 participants who used their visualization tool. The participants found that the tool helped them make better befriending decisions, especially when they received requests from fake accounts.

Wang et al. employed known concepts from behavioral decision research and soft paternalism in order to design mechanisms that "nudge" users to reconsider the content and context of their online disclosures before committing them [129]. The authors evaluated the effectiveness of their approach with 21 Facebook users in a three week exploratory field study and 13 follow-up interviews. Their results suggested that *privacy nudges* can be a promising way to prevent unintended disclosures when, for example, one befriends a fake or shares a post with the public.

4.3 Leveraging Victim Prediction

In Chapter 3, we introduced a new and complementary approach to thwart fake accounts in OSNs. In particular, we proposed to predict the potential victim of unknown fakes, and then leverage this information in existing defense mechanisms. In fact, Íntegro is one example of an application of this approach to graph-based fake account detection.

We herein take the position that *leveraging victim prediction represents a new security paradigm by itself*. To highlight the implication of this paradigm to existing defense mechanisms, we study two more security mechanisms that are widely-used to defend against fake accounts in OSNs; namely, user-facing security advice

(§4.3.1) and honeypots (§4.3.2). In what follows, we discuss how each mechanism can be improved by incorporating victim account prediction into its workflow.

4.3.1 User-Facing Security Advice

User education is the first line of defense against increasingly sophisticated social engineering attacks, especially in OSNs [58, 107]. While many studies showed that users tend to reject security advice because of low motivation and poor understanding of involved threats [1, 76], others asserted that users do so because it is entirely rational from an economic viewpoint [34, 48]. In particular, the advice offers to protect the users from the direct costs of attacks, but burdens them with increased indirect costs in the form of effort. When the security advice is applied to all users, it becomes a daily burden whose benefit is the potential saving of direct costs to potential victims; the fraction that might become victims. *When this fraction is small, designing a security advice that is beneficial becomes very hard*. For example, it is not feasible to burden 1.2 billion Facebook users with a daily task in order to spare 1% of them.

One way to increase the benefit of a security advice is to make it more usable, which in effect reduces its indirect costs to users. This has been the focus of a growing community of usable security researchers who consider user education essential to securing socio-technical systems such as OSNs [18]. Another complementary way to reduce indirect costs is to display the security advice to only the fraction who might actually benefit from it.

We propose to achieve this reduction by displaying the security advice in an informed, targeted manner. In particular, victim prediction provides an OSN with a robust way to quantify how vulnerable each real account is, as some users are more likely to be victims than others. The OSN can use this information to focus only on the most vulnerable population, and accordingly, educate them using a security advice while relieving the rest of the population from associated efforts.

4.3.2 Honeypots and User Sampling

Honeypots are accounts specially created or controlled to sample the activities of user accounts, in particular, those who contact these honeypots by sending them friend requests or by sharing content [108]. The sampled activities are then analyzed and used to maintain an up-to-date ground truth for fake account detection. While honeypot accounts are often used by third parties, OSNs still perform similar sampling albeit with direct access to user data [104]. Such a sampling technique, however, is inefficient as it is opportunistic if not completely random. For example, Stringhini et al. used 300 honeypot accounts in Facebook to record user activities over 12 months [108]. The collected dataset, however, was very small relatively to the sampling period, with only 3,831 friend requests (4.5% fake) and 72,431 messages (5.4% fake). To some extent, this also suggests that attackers do not necessarily target users at random.

Assuming that the percentage of fakes in the OSN is small, it is reasonable to sample the users at random and expect to collect mostly benign content originating from real accounts. The problem, however, is *when one samples for abusive content, as the sampling has to be biased towards unknown fake accounts*. For example, Facebook has more than 600 million daily active users and they perform billions of actions everyday [94]. In contrast, the number of fake accounts involved in an attack campaign is often on the order of thousands [16]. It is thus desired to have a reliable way to inform the user sampling process.

As victims are connected to abusive fakes, we propose to achieve this property by identifying potential victims of unknown fakes and then sampling the activities of their friends. Along with uniform random sampling [99], an OSN can achieve a desirable benign-to-abusive content ratio, which is important for effective feature-based detection using machine learning techniques [47].

Chapter 5

Impact and Conclusion

The ease with which we adopt online personas and relationships has created a soft spot that cyber criminals are willing to exploit. Advances in artificial intelligence make it feasible to design "socialbots" that sense, think, and act cooperatively in social settings just as in social robotics. In the wrong hands, these socialbots can be used to infiltrate online communities and carry out various malicious activities. From a computer security perspective, the concept of malicious socialbots is both interesting and disturbing, for the threat is no longer from a human controlling or monitoring a computer, but from exactly the opposite.

While the motivations for operating socialbots and the technical mechanisms that enable them remain rich areas of research, this dissertation focused on two research goals, which naturally divided the presentation into two parts:

1. *Threat characterization:* To understand and characterize what makes OSNs vulnerable to cyber-attacks by malicious socialbots. In particular, we focus on social infiltration at scale, in which malicious socialbots are used to connect with a large number of legitimate users, in order to mount subsequent attacks in the target OSN.

2. *Countermeasure design:* To design a new countermeasure to effectively and efficiently defend against socialbots that infiltrate users on a large scale. In

particular, the countermeasure has to be robust against social infiltration and scale to large OSNs consisting at least hundreds of millions of users.

In the fist part, we considered profit-driven attackers who infiltrate OSNs using malicious socialbots. Our investigation was guided by the following RQs:

- **RQ1:** How vulnerable are OSNs to social infiltration?
- **RQ2:** What are the security and privacy implications of social infiltration?
- **RQ3:** What is the economic rationale behind social infiltration at scale?

To address these questions, we studied social infiltration as an organized campaign run by a network of socialbots. We adopted the design of web-based botnets and defined what we call a socialbot network (SbN)—a group of programmable socialbots that are orchestrated by an attacker. We implemented an SbN prototype consisting of 100 socialbots and operated it on Facebook for 8 weeks in early 2011. The main findings of this study are:

1. OSNs such as Facebook suffer from inherent vulnerabilities that enable an attacker to automate social infiltration on a large scale.
2. Some users are more likely to become victims than others, which partly depends on factors related to their social structure.
3. Operating an SbN can result in serious privacy breaches, where personally identifiable information is compromised.
4. Traditional OSN defenses are not effective at identifying automated fake accounts nor their social infiltration campaigns.
5. In an economic context, an SbN ought to have a fixed size in terms of number of socialbots for it to be scalable in terms of number of attacked users.
6. Operating an SbN at scale is expected to be profitable, but it is not particularly attractive as an independent business.

These findings resulted in a timely public education about the threat [56], and has encouraged other researchers to replicate and extend this study on other OSNs such as Twitter and LinkedIn [29, 125]. The main contributions of this part of the dissertation are the following:

1. Demonstrating the feasibility of social infiltration in OSNs.

2. Economic analysis of social infiltration at scale.

In the second part, we built on top of earlier findings and considered attackers who can run a social infiltration campaign at a large scale using a set of automated fake accounts, or socialbots. We aimed to tackle the following question:

- **RQ4:** How can OSNs detect fakes that infiltrate users on a large scale?

To address this question, we designed Íntegro—a robust and scalable defense system that helps OSNs detect automated fake accounts via a user ranking scheme. Our design follows from previous findings, which shed light on the possibility of predicting potential victims of fakes in OSNs. In particular, Íntegro uses supervised machine learning with features extracted from user-level activities in order to identify potential victims in the OSN. By weighting the graph such that edges incident to potential victims have lower weights than other accounts, Íntegro guarantees that most real accounts are ranked higher than fakes. These ranks are derived from the landing probability of a modified random walk that starts from a known real account. To our knowledge, Íntegro is the first detection system that is robust against social infiltration, where fakes follow an adversarial strategy to befriend a large number of accounts, real or fake, in order to evade detection.

We implemented Íntegro on top of widely-used distributed systems, in which it scaled nearly linearly. We also evaluated Íntegro against SybilRank using our open-source comparative evaluation framework under real-world datasets and a production-class deployment at Tuenti. Our evaluation results showed that Íntegro significantly outperforms SybilRank in ranking quality, with up to an order of magnitude better precision in fake account detection. This part of the dissertation makes the following contributions:

1. Leveraging victim classification for fake account detection.

2. Open-source implementation and evaluation framework.

Íntegro is currently used in production at Tuenti to thwart fakes in the wild with at least 10 times higher precision, along side a proprietary feature-based detection system and a user-based abuse reporting system. It is also publicly released as part of an open-source project that is used by OSNs such as Facebook and Twitter.

Finally, we demonstrated that trying to prevent the threat of malicious socialbots leads to a set of hard socio-technical challenges that relate to web automation, online-offline identity binding, and usable security. Therefore, an OSN has more leverage in detecting socialbots and limiting their abuse as early as possible, rather than preventing their operation in the first place. In terms of future research, we highlighted account admission control as one possible direction, albeit it suffers from usability issues that discourage OSN operators from deploying it in practice. Instead, we advocated that leveraging victim prediction in OSNs is a new security paradigm, which can benefit different defense mechanisms, such as educating users with a user-facing security advice or detecting fakes using OSN honeypots.

Bibliography

[1] A. Adams and M. A. Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999. → pages 108

[2] L. Alvisi, A. Clement, A. Epasto, S. Lattanzi, and A. Panconesi. SoK: The evolution of sybil defense via social networks. In *Proceedings of the 34th IEEE Symposium on Security and Privacy*, pages 382–396. IEEE, 2013. → pages 5, 58, 62, 76, 80

[3] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker. Spamscatter: Characterizing internet scam hosting infrastructure. In *USENIX Security Symposium*, 2007. → pages 21

[4] R. Anderson. *Security engineering*. John Wiley & Sons, 2008. → pages 6, 17

[5] E. Behrends. *Introduction to Markov chains with special emphasis on rapid mixing*, volume 228. Vieweg, 2000. → pages 72, 132

[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 *Proceedings of the 18th international conference on World wide web*, pages 551–560. ACM, 2009. → pages 15, 17, 18, 21, 23, 58, 63

[7] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), 2008. → pages 75

[8] J. Bollen, H. Mao, and X. Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011. → pages 3, 15

[9] 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'09 Extended Abstracts on Human Factors in Computing Systems*, pages 2767–2770. ACM, 2009. → pages 32, 70

[10] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu. The socialbot network: when bots socialize for fame and money. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 93–102. ACM, 2011. → pages 35, 51, 53, 58, 69, 87, 94

[11] D. M. Boyd and N. B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1): 210–230, 2008. → pages 1, 3

[12] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. → pages 69, 83, 86

[13] G. Brown, T. Howe, M. Ihbe, A. Prakash, and K. Borders. Social networks and context-aware spam. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 403–412. ACM, 2008. → pages 18

[14] J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring pay-per-install: The commoditization of malware distribution. In *USENIX Security Symposium*, 2011. → pages 15, 16, 21

[15] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *NSDI*, pages 197–210, 2012. → pages 4, 12, 14, 41, 57, 61, 62, 63, 65, 77, 78, 79, 87, 95, 98

[16] Q. Cao, X. Yang, J. Yu, and C. Palow. Uncovering large groups of active malicious accounts in online social networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS'14, pages 477–488. ACM, 2014. → pages 61, 109

[17] CBC. Facebook shares drop on news of fake accounts, Aug 2012. URL http://goo.gl/6s5FKL. → pages 4, 57

[18] L. F. Cranor. *Security and usability: designing secure systems that people can use*. O'Reilly Media, Inc., 2005. → pages 105, 108

[19] G. Danezis and P. Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *Proceedings of the 9th Annual Network & Distributed System Security Symposium*. ACM, 2009. → pages 12, 41, 79

[20] J. B. Davies, A. Shorrocks, S. Sandstrom, and E. N. Wolff. The world distribution of household wealth. *Center for Global, International and Regional Studies*, 2007. → pages 19

[21] M. Dellamico and Y. Roudier. A measurement of mixing time in social networks. In *Proceedings of the 5th International Workshop on Security and Trust Management, Saint Malo, France*, 2009. → pages 72

[22] J. R. Douceur. The Sybil attack. In *Peer-to-peer Systems*, pages 251–260. Springer, 2002. → pages 2, 17, 104

[23] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology—CRYPTO'92*, pages 139–147. Springer, 1993. → pages 18

[24] D. Easley and J. Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010. → pages 1, 26, 31

[25] M. Egele, L. Bilge, E. Kirda, and C. Kruegel. Captcha smuggling: hijacking web browsing sessions to create captcha farms. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1865–1870. ACM, 2010. → pages 22

[26] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna. Compa: Detecting compromised accounts on social networks. In *NDSS*, 2013. → pages 59

[27] 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, 2007. → pages 32

[28] N. B. Ellison et al. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1): 210–230, 2007. → pages 15, 59

[29] A. Elyashar, M. Fire, D. Kagan, and Y. Elovici. Homing socialbots: intrusion on a specific organization's employee using socialbots. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 1358–1365. ACM, 2013. → pages 58, 63, 112

[30] Facebook. Investors relations: Annual earnings report. http://investor.fb.com, October 2014. → pages 1, 32, 52

[31] Facebook. Whitehat program: Reporting security vulnerabilities. https://facebook.com/whitehat/report, October 2014. → pages 33

[32] Facebook. Quarterly earning reports, Jan 2014. URL http://goo.gl/YujtO. → pages 57

[33] Facebook. Terms of service. https://www.facebook.com/legal/terms, May 2015. → pages 3, 4

[34] D. Florêncio and C. Herley. Where do security policies come from? In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, page 10. ACM, 2010. → pages 108

[35] D. Florêncio and C. Herley. Where do all the attacks go? In *Economics of Information Security and Privacy III*, pages 13–33. Springer, 2013. → pages 19, 55

[36] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3): 75–174, 2010. → pages 62

[37] M. Fossi, E. Johnson, D. Turner, T. Mack, J. Blackbird, D. McKinney, M. K. Low, T. Adams, M. P. Laucht, and J. Gough. Symantec report on the underground economy. *Symantec Corporation*, 2008. → pages 51, 52

[38] J. Franklin, A. Perrig, V. Paxson, and S. Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *ACM conference on Computer and communications security*, pages 375–388, 2007. → pages 15

[39] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in Facebook: A case study of unbiased sampling of osns. In *Proceedings of the 2010 IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2010. → pages 9, 34

[40] G. H. Golub and H. A. Van der Vorst. Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1): 35–65, 2000. → pages 70, 130

[41] M. Goncharov. Russian underground 101. *Trend Micro Incorporated Research Paper*, 2012. → pages 51

[42] C. Grier, L. Ballard, J. Caballero, N. Chachra, C. J. Dietrich, K. Levchenko, P. Mavrommatis, D. McCoy, A. Nappa, A. Pitsillidis, et al. Manufacturing compromise: the emergence of exploit-as-a-service. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 821–832. ACM, 2012. → pages 21

[43] J. Grimmelmann. Saving facebook. *Iowa Law Review*, 94:1137–1206, 2009. → pages 4

[44] R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80. ACM, 2005. → pages 40

[45] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 576–587. VLDB Endowment, 2004. → pages 70

[46] E. Hargittai et al. Facebook privacy settings: Who cares? *First Monday*, 15(8), 2010. → pages 20

[47] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: Data mining, inference, and prediction, second edition*. Springer, 2009. → pages 69, 70, 82, 83, 85, 86, 87, 109

[48] C. Herley. So long, and no thanks for the externalities: the rational rejection of security advice by users. In *Proceedings of the 2009 workshop on New security paradigms workshop*, pages 133–144. ACM, 2009. → pages 108

[49] C. Herley. The plight of the targeted attacker in a world of scale. In *The 9th Workshop on the Economics of Information Security*. ACM, 2010. → pages 18, 19, 22, 44, 54

[50] C. Herley and D. Florêncio. Nobody sells gold for the price of silver: Dishonesty, uncertainty and the underground economy. In *Economics of Information Security and Privacy*, pages 33–53. Springer, 2010. → pages 43, 47, 55

[51] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004. → pages 93

[52] C. J. Hernandez-Castro and A. Ribagorda. Remotely telling humans and computers apart: An unsolved problem. In *iNetSec 2009–Open Research Problems in Network Security*, pages 9–26. Springer, 2009. → pages 22

[53] T. Holz, C. Gorecki, F. Freiling, and K. Rieck. Detection and mitigation of fast-flux service networks. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium*, 2008. → pages 16, 21

[54] M. Huber, S. Kowalski, M. Nohlberg, and S. Tjoa. Towards automating social engineering using social networking sites. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 3, pages 117–124. IEEE, 2009. → pages 18, 29

[55] M. Huber, M. Mulazzani, and E. Weippl. Who on earth is "mr. cypher": Automated friend injection attacks on social networking sites. In *Security and Privacy–Silver Linings in the Cloud*, pages 80–89. Springer, 2010. → pages 25

[56] T. Hwang, I. Pearce, and M. Nanis. Socialbots: Voices from the fronts. *interactions*, 19(2):38–45, 2012. → pages 1, 2, 5, 18, 59, 112

[57] D. Irani, M. Balduzzi, D. Balzarotti, E. Kirda, and C. Pu. Reverse social engineering attacks in online social networks. In *Detection of intrusions and malware, and vulnerability assessment*, pages 55–74. Springer, 2011. → pages 17, 18

[58] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, 2007. → pages 3, 18, 105, 108

[59] A. Jaimoukha. *Circassian Proverbs & Sayings*. Sanjalay Book Press, 2009. → pages iv

[60] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM, 2003. → pages 12, 41, 79

[61] C. Kanich, N. Weaver, D. McCoy, T. Halvorson, C. Kreibich, K. Levchenko, V. Paxson, G. M. Voelker, and S. Savage. Show me the money: Characterizing spam-advertised revenue. In *USENIX Security Symposium*, 2011. → pages 55

[62] 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. → pages 1

[63] E. J. Kartaltepe, J. A. Morales, S. Xu, and R. Sandhu. Social network-based botnet command-and-control: emerging threats and countermeasures. In *Applied Cryptography and Network Security*, pages 511–528. Springer, 2010. → pages 15, 32

[64] G. Karypis and V. Kumar. Multilevel *k*-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing*, 48(1): 96–129, 1998. → pages 61

[65] H. Kim, J. Tang, and R. Anderson. Social authentication: harder than it looks. In *Financial Cryptography and Data Security*, pages 1–15. Springer, 2012. → pages 103

[66] T. H.-J. Kim, A. Yamada, V. Gligor, J. Hong, and A. Perrig. Relationgram: Tie-strength visualization for user-controlled online identity authentication. In *In Proceedings of Financial Cryptography and Data Security Conference*, pages 69–77. Springer, 2013. → pages 107

[67] M. N. Ko, G. P. Cheek, M. Shehab, and R. Sandhu. Social-networks connect services. *Computer*, 43(8):37–43, 2010. → pages 24, 34

[68] 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*, pages 721–730. ACM, 2008. → pages 20, 32

[69] T. Lauinger, V. Pankakoski, D. Balzarotti, and E. Kirda. Honeybot, your man in the middle for automated social engineering. In *Proceedings of the*

3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats. USENIX Association, 2010. → pages 29

[70] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2006. → pages 81

[71] J. Leskovec and E. Horvitz. Planetary-scale views on a large instant-messaging network. In *Proceedings of the 17th international conference on World Wide Web*, pages 915–924. ACM, 2008. → pages 31, 74

[72] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009. → pages 62, 72, 74

[73] Y. Liu, K. P. Gummadi, B. Krishnamurthy, and A. Mislove. Analyzing facebook privacy settings: user expectations vs. reality. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 61–70. ACM, 2011. → pages 21, 105

[74] D. Lowd and C. Meek. Adversarial learning. In *Proceedings of the 11th ACM International conference on Knowledge Discovery in Data Mining*, pages 641–647. ACM, 2005. → pages 5, 28, 35, 61

[75] E. Maler and D. Reed. The venn of identity. *IEEE Security and Privacy*, 6 (2):16–23, 2008. → pages 105

[76] M. Mannan and P. C. van Oorschot. Security and usability: the gap in real-world online banking. In *Proceedings of the 2007 Workshop on New Security Paradigms*, pages 1–14. ACM, 2008. → pages 108

[77] D. McCoy, A. Pitsillidis, G. Jordan, N. Weaver, C. Kreibich, B. Krebs, G. M. Voelker, S. Savage, and K. Levchenko. Pharmaleaks: Understanding the business of online pharmaceutical affiliate programs. In *Proceedings of the 21st USENIX Conference on Security Symposium*. USENIX Association, 2012. → pages 15, 21

[78] T. Merrill, K. Latham, R. Santalesa, and D. Navetta. Social media: The business benefits may be enormous, but can the risks–reputational, legal, operational–be mitigated? *ACE Group*, 2011. → pages 3

[79] D. Misener. Rise of the socialbots: They could be influencing you online. http://goo.gl/TX7c1p, March 2011. → pages 2

[80] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42. ACM, 2007. → pages 24, 54

[81] A. Mislove, A. Post, P. Druschel, and P. K. Gummadi. Ostra: Leveraging trust to thwart unwanted communication. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 15–30. USENIX Association, 2008. → pages 106

[82] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: inferring user profiles in online social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 251–260. ACM, 2010. → pages 62, 80

[83] A. Mohaisen, A. Yun, and Y. Kim. Measuring the mixing time of social graphs. In *Proceedings of the 10th annual conference on Internet measurement*, pages 383–389. ACM, 2010. → pages 72

[84] A. Mohaisen, H. Tran, N. Hopper, and Y. Kim. On the mixing time of directed social graphs and security implications. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 36–37. ACM, 2012. → pages 98

[85] M. Mondal, B. Viswanath, A. Clement, P. Druschel, K. P. Gummadi, A. Mislove, and A. Post. Limiting large-scale crawls of social networking sites. *ACM SIGCOMM Computer Communication Review*, 41(4): 398–399, 2011. → pages 103, 106

[86] T. Moore, R. Clayton, and R. Anderson. The economics of online crime. *The Journal of Economic Perspectives*, 23(3):3–20, 2009. → pages 15

[87] 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 2010 USENIX Security Symposium*, volume 10, pages 4–1. USENIX Association, 2010. → pages 16, 21, 22, 102

[88] 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 conference on Security*, pages 14–14. USENIX Association, 2011. → pages 15, 21, 22, 51, 53, 59

[89] R. Mourtada and F. Salem. Civil movements: The impact of facebook and twitter. *Arab Social Media Report*, 1(2):1–30, 2011. → pages 1

[90] S. Nagaraja, A. Houmansadr, P. Piyawongwisal, V. Singh, P. Agarwal, and N. Borisov. Stegobot: a covert social network botnet. In *Information Hiding*, pages 299–313. Springer, 2011. → pages 32

[91] F. Nagle and L. Singh. Can friends be trusted? exploring privacy in online social networks. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Network Analysis and Mining*, pages 312–315. IEEE, 2009. → pages 17, 31

[92] M. Nanis, I. Pearce, and T. Hwang. Pacific social architecting corporation: Field test report. http://pacsocial.com, November 2011. → pages 18

[93] M. E. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006. → pages 75

[94] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, D. Stafford, T. Tung, and V. Venkataramani. Scaling memcache at Facebook. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, NSDI'13, pages 385–398. USENIX Association, 2013. → pages 109

[95] J. Nolan and M. Levesque. Hacking human: data-archaeology and surveillance in social networks. *ACM SIGGROUP Bulletin*, 25(2):33–37, 2005. → pages 3, 15, 18, 40

[96] R. Potharaju, B. Carbunar, and C. Nita-Rotaru. iFriendU: Leveraging 3-cliques to enhance infiltration attacks in online social networks. In

Proceedings of the 17th ACM conference on Computer and communications security, pages 723–725. ACM, 2010. → pages 17

[97] H. Rashtian, Y. Boshmaf, P. Jaferian, and K. Beznosov. To befriend or not? a model of friend request acceptance on facebook. In *Symposium on Usable Privacy and Security (SOUPS)*, 2014. → pages 105

[98] J. Ratkiewicz, M. Conover, M. Meiss, B. Gonçalves, A. Flammini, and F. Menczer. Detecting and tracking political abuse in social media. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media*, 2011. → pages 3

[99] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, 2005. → pages 34, 109

[100] M. Scherer. Obama's 2012 digital fundraising outperformed 2008. http://goo.gl/pvYiQh, November 2012. → pages 1

[101] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles. Botnets: A survey. *Computer Networks*, 57(2):378–403, 2013. → pages 7, 26, 33

[102] A. Sinclair. Improved bounds for mixing rates of Markov chains and multicommodity flow. In *Proceedings of the 1st Latin American Symposium on Theoretical Informatics*, pages 474–487. Springer-Verlag, 1992. → pages 77, 131

[103] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2004. → pages 65

[104] T. Stein, E. Chen, and K. Mangla. Facebook immune system. In *Proceedings of the 4th Workshop on Social Network Systems*, page 8. ACM, 2011. → pages 4, 5, 9, 17, 32, 55, 57, 58, 60, 63, 98, 103, 106, 109

[105] B. Stone-Gross, R. Abman, R. A. Kemmerer, C. Kruegel, D. G. Steigerwald, and G. Vigna. The underground economy of fake antivirus software. In *Economics of Information Security and Privacy III*, pages 55–78. Springer, 2013. → pages 15, 21

[106] L. J. Strahilevitz. A social networks theory of privacy. *The University of Chicago Law Review*, pages 919–988, 2005. → pages 4

[107] K. Strater and H. R. Lipford. Strategies and struggles with privacy in an online social networking community. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction-Volume 1*, pages 111–119. British Computer Society, 2008. → pages 108

[108] G. Stringhini, C. Kruegel, and G. Vigna. Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 1–9. ACM, 2010. → pages 4, 60, 109

[109] G. Stringhini, G. Wang, M. Egele, C. Kruegel, G. Vigna, H. Zheng, and B. Y. Zhao. Follow the green: growth and dynamics in twitter follower markets. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 163–176. ACM, 2013. → pages 43, 56, 64

[110] S.-T. Sun, Y. Boshmaf, K. Hawkey, and K. Beznosov. A billion keys, but few locks: the crisis of web single sign-on. In *Proceedings of the 2010 workshop on New security paradigms*, pages 61–72. ACM, 2010. → pages 105

[111] D. Thibeau and D. Reed. Open trust frameworks for open government: Enabling citizen involvement through open identity technologies. *White paper, OpenID Foudation and Information Card Foudation*, 2009. → pages 105

[112] K. Thomas and D. M. Nicol. The Koobface botnet and the rise of social malware. In *Proceedings of the 5th International Conference on Malicious and Unwanted Software*, pages 63–70. IEEE, 2010. → pages 3, 102, 103

[113] K. Thomas, C. Grier, and V. Paxson. Adapting social spam infrastructure for political censorship. In *Proceedings of the 5th USENIX workshop on Large-Scale Exploits and Emergent Threats*, pages 13–23. USENIX Association, 2012. → pages 3, 15

[114] K. Thomas, D. McCoy, C. Grier, A. Kolcz, and V. Paxson. Trafficking fraudulent accounts: The role of the underground market in twitter spam and abuse. In *Proceedings of the 22nd USENIX Conference on Security*, pages 195–210. USENIX Association, 2013. → pages 3, 20, 21, 43, 51, 56

[115] Threat landscape research lab. Fortinet report on the anatomy of a botnet. *Fortinet Corporation*, 2013. → pages 52

[116] 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. → pages 30

[117] D. N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-resilient online content voting. In *NSDI*, volume 9, pages 15–28, 2009. → pages 80

[118] N. Tran, J. Li, L. Subramanian, and S. S. Chow. Optimal sybil-resilient node admission control. In *INFOCOM, 2011 Proceedings IEEE*, pages 3218–3226. IEEE, 2011. → pages 12, 41, 79, 80

[119] Twitter. Investors relations: Annual earnings report. http://investor.twitterinc.com, October 2014. → pages 1

[120] J. Tygar. Adversarial machine learning. *IEEE Internet Computing*, 15(5), 2011. → pages 61, 69

[121] H. R. Varian and W. Norton. *Microeconomic analysis*, volume 2. Norton New York, 1992. → pages 46, 47, 54, 55

[122] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove. An analysis of social network-based sybil defenses. In *Proceedings of ACM SIGCOMM Computer Communication Review*, pages 363–374. ACM, 2010. → pages 12, 13, 62, 65, 75, 79, 80

[123] B. Viswanath, M. Mondal, A. Clement, P. Druschel, K. P. Gummadi, A. Mislove, and A. Post. Exploring the design space of social network-based sybil defenses. In *In proceedings of the 4th International Conference on Communication Systems and Networks*, pages 1–8. IEEE, 2012. → pages 5, 62

[124] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In *Advances in Cryptology—EUROCRYPT 2003*, pages 294–311. Springer, 2003. → pages 21, 102

[125] C. Wagner, S. Mitter, C. Körner, and M. Strohmaier. When social bots attack: Modeling susceptibility of users in online social networks. In *Proceedings of the WWW*, volume 12, 2012. → pages 58, 63, 64, 112

[126] G. Wang, C. Wilson, X. Zhao, Y. Zhu, M. Mohanlal, H. Zheng, and B. Y. Zhao. Serf and turf: crowdturfing for fun and profit. In *Proceedings of the 21st international conference on World Wide Web*, pages 679–688. ACM, 2012. → pages 21

[127] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Y. Zhao. You are how you click: Clickstream analysis for sybil detection. In *Proceedings of the 22nd USENIX Conference on Security*, pages 241–256. USENIX Association, 2013. → pages 4, 60

[128] G. Wang, M. Mohanlal, C. Wilson, X. Wang, M. Metzger, H. Zheng, and B. Y. Zhao. Social turing tests: Crowdsourcing sybil detection. In *Proceedings of the 20th Annual Network & Distributed System Security Symposium*. ACM, 2013. → pages 63

[129] Y. Wang, P. G. Leon, K. Scott, X. Chen, A. Acquisti, and L. F. Cranor. Privacy nudges for social media: an exploratory facebook study. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 763–770. International World Wide Web Conferences Steering Committee, 2013. → pages 107

[130] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *nature*, 393(6684):440–442, 1998. → pages 87, 96

[131] Y. Xie, F. Yu, Q. Ke, M. Abadi, E. Gillum, K. Vitaldevaria, J. Walter, J. Huang, and Z. M. Mao. Innocent by association: early recognition of legitimate users. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 353–364. ACM, 2012. → pages 99, 106

[132] 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*, pages 196–206. ACM, 2011. → pages 3, 15

[133] C. Yang, R. Harkreader, J. Zhang, S. Shin, and G. Gu. Analyzing spammers' social networks for fun and profit: a case study of cyber criminal ecosystem on twitter. In *Proceedings of the 21st international conference on World Wide Web*, pages 71–80. ACM, 2012. → pages 64

[134] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai. Uncovering social network sybils in the wild. In *Proceedings of the 2011 ACM SIGCOMM Internet Measurement Csonference*, pages 259–268. ACM, 2011. → pages 60, 63, 76

[135] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai. Uncovering social network sybils in the wild. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(1):2, 2014. → pages 4, 30

[136] S. Yardi, N. Feamster, and A. Bruckman. Photo-based authentication using social networks. In *Proceedings of the first workshop on Online social networks*, pages 55–60. ACM, 2008. → pages 63, 103

[137] H. Yu. Sybil defenses via social networks: a tutorial and survey. *ACM SIGACT News*, 42(3):80–101, 2011. → pages 3, 5, 30, 41, 58, 62, 65, 87, 98

[138] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. *ACM SIGCOMM Computer Communication Review*, 36(4):267–278, 2006. → pages 12, 62, 79

[139] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 3–17. IEEE, 2008. → pages 12, 41, 62, 79

[140] C. M. Zhang and V. Paxson. Detecting and analyzing automated activity on twitter. In *Passive and Active Measurement*, pages 102–111. Springer, 2011. → pages 104

Appendix A

Security Analysis of Íntegro

In what follows, we provide the required background on random walks after which we analyze the main security guarantee of Íntegro.

A.1 Background

Let $G = (V, E)$ be an undirected graph with $n = |V|$ nodes and $m = |E|$ undirected edges. Also, let $w : E \to \mathbb{R}^+$ be a function that assigns each edge $(v_i, v_j) \in E$ a weight $w(v_i, v_j) > 0$. The *transition matrix* P is an $n \times n$ matrix, where each entry $p_{ij} \in [0, 1]$ represents the probability of moving from node $v_i \in V$ to node $v_j \in V$, as defined by:

$$p_{ij} := \begin{cases} \dfrac{w(v_i, v_j)}{\deg(v_i)} & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{A.1}$$

The transition matrix P might not be symmetric but it is right-stochastic, as P is a square matrix of non-negative real numbers and for each node $v_i \in V$,

$$\sum_{(v_i, v_j) \in E} p_{ij} = 1. \tag{A.2}$$

The event of moving from one node to another in G is captured by a *Markov*

chain representing a random walk over G. In turn, a random walk $W = \langle v_1, \ldots, v_i \rangle$ of length $i \geq 1$ over G is a sequence of nodes that starts at the initial node v_1 and ends at the terminal node v_i, following the transition probability defined in Equation A.1. The Markov chain is called *ergodic* if it is irreducible and aperiodic. In this case, the Markov chain has a unique stationary distribution to which the random walk converges as $i \to \infty$. The *stationary distribution* π of a Markov chain is a probability distribution that is invariant to the transition matrix, that is, whenever $\pi P = \pi$. The stationary distribution of the Markov chain over G is a $1 \times n$ probability vector, and is defined by:

$$\pi := \left[\frac{\deg(v_1)}{\mathrm{vol}(V)} \ \cdots \ \frac{\deg(v_n)}{\mathrm{vol}(V)} \right], \tag{A.3}$$

where $\pi(v_j)$ is the jth entry in π and represents the landing probability of node $v_j \in V$, and $\mathrm{vol}(U)$ is the volume of a node set $U \subseteq V$, which is defined by:

$$\mathrm{vol}(U) := \sum_{v_j \in U} \deg(v_j). \tag{A.4}$$

The *marginal distribution* π_i of the Markov chain over G is a $1 \times n$ probability vector, where $\pi_i(v_j)$ is the landing probability of node $v_j \in V$ at step i of the random walk. Given an initial distribution π_0, the marginal distribution π_i can be iteratively defined by:

$$\pi_i := \pi_{i-1} P = \pi_0 P^i, \tag{A.5}$$

and accordingly, $\pi_i(v_j)$ can be computed by [40]:

$$\pi_i(v_j) = \sum_{(v_k, v_j) \in E} \pi_{i-1}(v_k) \cdot \frac{w(v_k, v_j)}{\deg(v_k)}. \tag{A.6}$$

The *total variation* distance $||\pi_i - \pi||_{\mathrm{TV}}$ between the marginal and stationary

distributions is a measure of how "close" these distribution are, and is defined by

$$||\pi_i - \pi||_{\text{TV}} := \frac{1}{2} \sum_{v_j \in V} |\pi_i(v_j) - \pi(v_j)|. \tag{A.7}$$

The *mixing time* $\mathcal{T}(\varepsilon)$ of the Markov chain over G, when parametrized by a *relative variation error* $\varepsilon > 0$, is the minimal length of the random walk required for the marginal distribution to be ε-close to the stationary distribution in total variation distance, and is defined by

$$\mathcal{T}(\varepsilon) := \min \{ i : ||\pi_i - \pi||_{\text{TV}} \leq \varepsilon \}. \tag{A.8}$$

It thus follows that if $i \geq \mathcal{T}(\varepsilon)$, we have $\pi_i = \pi$.

A.2 Mathematical Proofs

We next start by proving that reassigning edge weights in an undirected graph changes its mixing time by only a constant factor. We subscript the used notation in order to differentiate between different graphs when necessary. For a given OSN, *we refer to its social graph after rate adjustment as the defense graph D.*

Lemma 1: *Given a social graph G with a mixing time $\mathcal{T}_G(\varepsilon)$, the corresponding defense graph D after rate adjustment has a mixing time $\mathcal{T}_D(\varepsilon) = O(\mathcal{T}_G(\varepsilon))$.*

Proof. Recall that the mixing time of an undirected graph $G = (V, E)$ is bounded by [102]:

$$\frac{\lambda}{2(1-\lambda)} \log\left(\frac{1}{2\varepsilon}\right) \leq \mathcal{T}(\varepsilon) \leq \frac{\log(n) + \log\left(\frac{1}{\varepsilon}\right)}{1-\lambda}, \tag{A.9}$$

where $\lambda \in (-1, 1)$ is the second largest eigenvalue of the transition matrix P of the graph G. For a social graph G and its defense graph D, we have

$$\frac{\mathcal{T}_D(\varepsilon)}{\mathcal{T}_G(\varepsilon)} \leq \frac{1-\lambda_G}{1-\lambda_D} = O(1),$$

and thus, $\mathcal{T}_D(\varepsilon) = O(\mathcal{T}_G(\varepsilon))$. □

Given the bound in Equation A.9, a Markov chain over a graph G is *fast mixing* if $\mathcal{T}(\varepsilon)$ is polynomial in $\log n$ and $\log(1/\varepsilon)$. In the context of fake account detection, we consider the stricter case when $\varepsilon = O(1/n)$, and so we call G fast mixing if $\mathcal{T}(\varepsilon) = O(\log n)$.

Let us refer to the landing probability $\pi_i(v_j)$ of a node $v_j \in V$ as its *trust value* so that π_i is the trust distribution in step i.[1] Moreover, let the *expansion* $\chi(U)$ of a node set $U \subseteq V$ be defined by

$$\chi(U) := \frac{\text{vol}\,(\partial(U))}{\text{vol}(U)}, \tag{A.10}$$

where $\partial(U) = \{(v_i, v_j) \in E : v_i \in U,\ v_j \in V \setminus U\}$ is the *edge boundary* of U. In our threat model, we have $\partial(V_r) = \partial(V_f) = E_a$, where *edges in E_a are established at random*.

We now prove that during a random walk on the defense graph $D = (V, E)$, where the walk starts from a known real node picked at random in the real region, *the expected aggregate trust in the fake region D_f monotonically increases by diminishing increments until it converges to its stationary value*.

Lemma 2: *Given a defense graph D with $g = |E_a|$ randomly established attack edges, $n_0 \geq 1$ trusted real nodes, a total trust $\tau \geq 1$, and an initial trust distribution*

$$\pi_0(v_j) = \begin{cases} \tau/n_0 & \text{if } v_j \text{ is a trusted node,} \\ 0 & \text{otherwise,} \end{cases}$$

the expected aggregate trust over the fake region in the $(i+1)$-th iterations increases by an amount of $(\chi(V_r) \cdot \tau)\left(1 - \chi(V_r) - \chi(V_f)\right)^i$ for each $i \geq 0$.

Proof. We prove the lemma by induction. We use the iterative method described in Equation A.5 to compute trust distribution for a random walk that starts from

[1] In Chapter 3, we denoted the trust value π_i by T_i. The reason we use π_i herein is because it is the standard notation used in analyzing stochastic processes [5].

a trusted node. We first define some notation. Let $\pi_i(V_r)$ be the aggregate trust in the real region D_r after iteration i, as defined by

$$\pi_i(V_r) := \sum_{v_j \in V_r} \pi_i(v_j). \tag{A.11}$$

Similarly, let $\pi_i(V_f)$ be the aggregate trust in D_f after iteration i. As defined by π_0, initially, we have $\pi_0(V_r) = \tau$ and $\pi_0(V_f) = 0$. Moreover, the total trust τ is reserved during the iterations, that is, $\pi_i(V_r) + \pi_i(V_f) = \tau$ for each $i \geq 0$.

In each iteration i, the total trust is redistributed in the graph. Consider iteration $i+1$. For each $v_i, v_j \in V_r$, the edge $(v_i, v_j) \in E$ carries $w(v_i, v_j)\,(\pi_i(V_r)/\mathrm{vol}(V_r))$ trust on average. As the $|\partial(V_r)|$ attack edges are established at random, it is expected that $\mathrm{vol}\,(\partial(V_r))\,(\pi_i(V_r)/\mathrm{vol}(V_r))$ trust, that is, $\chi(V_r) \cdot \pi_i(V_r)$, is passed through these edges to the fake region. The same also holds for the fake region, which means we can model the trust exchange between D_r and D_f by

$$\begin{aligned}
\pi_{i+1}(V_r) &= \pi_i(V_r) + \chi(V_f) \cdot \pi_i(V_f) - \chi(V_r) \cdot \pi_i(V_r) \text{ and} \\
\pi_{i+1}(V_f) &= \pi_i(V_f) + \chi(H) \cdot \pi_i(V_r) - \chi(V_f) \cdot \pi_i(V_f),
\end{aligned}$$

where the total trust τ is conserved throughout the process, as follows:

$$\pi_{i+1}(V_r) + \pi_{i+1}(V_f) = \pi_i(V_r) + \pi_i(V_f) = \tau.$$

We now consider the base case of this lemma. Initially, for iteration $i = 0$, we have $\chi(V_r) \cdot \pi_0(V_r) = \chi(V_r) \cdot \tau$ and $\chi(V_f) \cdot \pi_0(V_f) = 0$. Therefore,

$$\pi_1(V_f) - \pi_0(V_f) = \chi(V_r) \cdot \tau.$$

We next state the induction hypothesis. For each $i \geq 1$, let us assume the following statement is true:

$$\pi_i(V_f) - \pi_{i-1}(V_f) = (\chi(V_r) \cdot \tau)\left(1 - \chi(V_r) - \chi(V_f)\right)^{i-1}.$$

Now, let us consider the trust exchange in iteration $i+1$:

$$\pi_{i+1}(V_f) - \pi_i(V_f) = \chi(V_r) \cdot \pi_i(V_r) - \chi(V_f) \cdot \pi_i(V_f)$$

By substituting $\pi_i(V_r)$ by $\pi_{i-1}(V_r) + \chi(V_f) \cdot \pi_{i-1}(V_f) - \chi(V_r)\tau_{i-1}(V_r)$, and doing similarly so for $\pi_{i-1}(V_f)$, we get

$$\begin{aligned}\pi_{i+1}(V_f) - \pi_i(V_f) &= \chi(V_r)\left((1-\chi(V_r)) \cdot \pi_{i-1}(V_r) + \chi(V_f) \cdot \pi_{i-1}(V_f)\right) - \\ &\quad \chi(V_f)\left(\left(1-\chi(V_f)\right) \cdot \pi_{i-1}(V_f) + \chi(V_r) \cdot \pi_{i-1}(V_r)\right) \\ &= \left(\chi(V_r) \cdot \pi_{i-1}(V_r) - \chi(V_f) \cdot \pi_{i-1}(V_f)\right)\left(1-\chi(V_r)-\chi(V_f)\right).\end{aligned}$$

We know that $\pi_i(V_f) = \pi_{i-1}(V_f) + \chi(V_r) \cdot \pi_{i-1}(V_r) - \chi(V_f) \cdot \pi_{i-1}(V_f)$, and so

$$\pi_{i+1}(V_f) - \pi_i(V_f) = \left(\pi_i(V_f) - \pi_{i-1}(V_f)\right)\left(1-\chi(V_r)-\chi(V_f)\right).$$

Finally, by the induction hypothesis, we end up with

$$\pi_{i+1}(V_f) - \pi_i(V_f) = \left(\chi(V_r) \cdot \tau\right)\left(1-\chi(V_r)-\chi(V_f)\right)^i,$$

which, by induction, completes the proof. □

Corollary 2: *In the i-th iteration, the expected increment of aggregate trust in the fake region in upper bounded by* $(\chi(V_r) \cdot \tau)\,(1-\chi(V_r))^i$ *for each* $i \geq 0$.

We next bound the aggregate trust in the fake region $\pi_i(V_f)$ after β iterations, where $1 \leq \beta \leq \mathcal{T}(\varepsilon) - \Delta$ and $\Delta > 1$ is a positive natural number. We achieve this by directly comparing $\pi_\beta(V_f)$ to its stationary value $\pi_{\mathcal{T}(\varepsilon)}(V_f)$, where $\mathcal{T}(\varepsilon) \geq \beta + \Delta$. In fact, this result holds as long as there is at least a constant difference between the mixing time $\mathcal{T}(\varepsilon)$ and β, or in other words, whenever $\mathcal{T}(\varepsilon) - \beta = \Omega(1)$ and $\mathcal{T}(\varepsilon)$ is not arbitrarily large.

Lemma 3: *Given a defense graph D with a mixing time* $\mathcal{T}(\varepsilon) \geq 1$ *and a positive integer* $\beta \in [1, \mathcal{T}(\varepsilon) - \Delta]$ *where* $\Delta > 1$*, the aggregate trust in the fake region*

$\pi_\beta(V_f)$ after β iterations gets a fraction $f \in (0,1)$ of that in the stationary distribution, that is, $\pi_\beta(V_f) = f \cdot \tau \cdot \big(\mathrm{vol}(V_f)/\mathrm{vol}(V)\big)$, where

$$f = \frac{\beta \cdot \sum_{0 \le i \le \beta - \Delta} \big(1 - \chi(V_r) - \chi(V_f)\big)^i}{\mathcal{T}(\varepsilon) \cdot \sum_{0 \le i \le \mathcal{T}(\varepsilon) - \Delta} \big(1 - \chi(V_r) - \chi(V_f)\big)^i} \tag{A.12}$$

Proof. By Lemma 2, we know that the aggregate trust in the fake region monotonically increases with the number of iterations in the process defined by Equation A.5. For iteration $i = \beta$, where $1 \le \beta \le \mathcal{T}(\varepsilon) - \Delta$, we have

$$\begin{aligned}
\pi_\beta(V_f) &= \sum_{0 \le i \le \beta - \Delta} (\chi(V_r) \cdot \tau) \big(1 - \chi(V_r) - \chi(V_f)\big)^i \\
&= (\beta \cdot \chi(V_r) \cdot \tau) \sum_{0 \le i \le \beta - \Delta} \big(1 - \chi(V_r) - \chi(V_f)\big)^i.
\end{aligned}$$

Similarly, for iteration $i = \gamma$, where $\gamma = \mathcal{T}(\varepsilon) \ge \beta + \Delta$, we have

$$\begin{aligned}
\pi_\gamma(V_f) &= \sum_{0 \le i \le \gamma - \Delta} (\chi(V_r) \cdot \tau) \big(1 - \chi(V_r) - \chi(V_f)\big)^i \\
&= (\gamma \cdot \chi(V_r) \cdot \tau) \sum_{0 \le i \le \gamma - \Delta} \big(1 - \chi(V_r) - \chi(V_f)\big)^i.
\end{aligned}$$

Now let us consider the ratio $\pi_\beta(V_f)/\pi_\gamma(V_f)$. We have

$$\frac{\pi_\beta(V_f)}{\pi_\gamma(V_f)} = \frac{(\beta \cdot \chi(V_r) \cdot \tau) \sum_{0 \le i \le \beta - \Delta} \big(1 - \chi(V_f) - \chi(V_f)\big)^i}{(\gamma \cdot \chi(V_r) \cdot \tau) \sum_{0 \le i \le \gamma - \Delta} \big(1 - \chi(V_r) - \chi(V_f)\big)^i}.$$

By multiplying both side by $\pi_\gamma(V_f)$, we get

$$\pi_\beta(V_f) = \frac{\beta \cdot \sum_{0 \le i \le \beta - \Delta} \big(1 - \chi(V_r) - \chi(V_f)\big)^i}{\gamma \cdot \sum_{0 \le i \le \gamma - \Delta} \big(1 - \chi(V_r) - \chi(V_f)\big)^i} \cdot \pi_\gamma(V_f). \tag{A.13}$$

Now, recall that $\pi_\gamma(v_j) = \tau \cdot \pi(v_j) = \tau \cdot \big(\deg(v_j)/\mathrm{vol}(V)\big)$ for each $v_j \in V_f$, where

π is the stationary distribution of the graph D (see Equation A.3), Accordingly,

$$\begin{aligned}\pi_\beta(V_f) &= \frac{\beta \cdot \sum_{0\le i\le \beta-\Delta}\big(1-\chi(V_r)-\chi(V_f)\big)^i}{\gamma \cdot \sum_{0\le i\le \gamma-\Delta}\big(1-\chi(V_r)-\chi(V_f)\big)^i} \cdot \tau \cdot \frac{\mathrm{vol}(V_f)}{\mathrm{vol}(V)} \\ &= f \cdot \tau \cdot \frac{\mathrm{vol}(V_f)}{\mathrm{vol}(V)}\end{aligned}$$

Finally, as $\beta \le \gamma - \Delta$, we have $\beta/\gamma \le (\gamma-\Delta)/\gamma$. As γ is not arbitrarily large, $\beta/\gamma < 1$ holds. Therefore, $f < 1$. □

As the total trust τ is conserved, Corollary 3 below directly follows.

Corollary 3: *For a positive number $\Delta > 1$, if the aggregate trust in the fake region after $1 \le \beta \le \mathcal{T}(\varepsilon) - \Delta$ iterations is a fraction $f \in (0,1)$ of that in the stationary distribution, then the aggregate trust in the real region during the same iteration is $c > 1$ times of that in the stationary distribution, that is, $\pi_\beta(V_r) = c \cdot \tau \cdot (\mathrm{vol}(V_r)/\mathrm{vol}(V))$, where $c = 1 + (1-f)\big(\mathrm{vol}(V_f)/\mathrm{vol}(V_r)\big)$.*

Given a fraction $f > 1$ and a multiplier $c > 1$, as defined by Lemma 3 and Corollary 3, the trust distribution over nodes in D after β iterations is defined by

$$\pi_\beta(v_j) = \begin{cases} f \cdot \tau \cdot \dfrac{\deg(v_i)}{\mathrm{vol}(V)} < 1 & \text{if } v_j \in V_f, \\ c \cdot \tau \cdot \dfrac{\deg(v_i)}{\mathrm{vol}(V)} > 1 & \text{if } v_j \in V_r. \end{cases} \tag{A.14}$$

Moreover, let $\bar{\pi}_\beta(v_j) = \pi_\beta(v_j)/\deg(v_i)$ be the degree-normalized trust for each $v_j \in V$, as derived from Equation A.14. We next prove that at most $(f/c)\cdot \mathrm{vol}(V_f)$ fake nodes can have degree-normalized trust or *rank values* higher than or equal to $(c\cdot\tau)/\mathrm{vol}(V)$.

Lemma 4: *Consider a defense graph D with a mixing time $\gamma = \mathcal{T}(\varepsilon)$, a fraction $f < 0$, and a multiplier $c > 1$ such that $\pi_\beta(V_r) = c \cdot \pi_\gamma(V_r)$ and $\pi_\beta(V_f) = f \cdot \pi_\gamma(V_f)$ after $1 \le \beta \le \mathcal{T}(\varepsilon) - \Delta$ power iterations for some $\Delta > 1$. Regardless to how an attacker organizes the fake region, there can be only a set $U \subset V_f$ of at most*

$(f/c) \cdot \text{vol}(V_f)$ *fake nodes such that each* $v_j \in U$ *has a degree-normalized trust* $\bar{\pi}_\beta(v_j) \geq (c \cdot \tau)/\text{vol}(V)$.

Proof. For an attacker, the optimal strategy to maximize the cardinality of the set U is to assign $(c \cdot \tau)/\text{vol}(V)$ degree-normalized trust to as many fake nodes as possible, and then leave the rest of the fake nodes with zero trust.

We now prove by contradiction that $|U| \leq (f/c) \cdot \text{vol}(V_f)$. Assume the opposite, where $|U| > (f/c) \cdot \text{vol}(V_f)$. Since each node $v_j \in U$ is connected to at least another node $v_k \in U$, or otherwise it would be disconnected and $\bar{\pi}_\beta(v_j) = 0$, then the aggregate degree-normalized trust $\bar{\pi}_\beta(V_f)$ in the fake region is

$$\bar{\pi}_\beta(V_f) = |U| \cdot \frac{c \cdot \tau}{\text{vol}(V)} \tag{A.15}$$

$$> \frac{f}{c} \cdot \text{vol}(V_f) \cdot \frac{c \cdot \tau}{\text{vol}(V)} \tag{A.16}$$

$$> f \cdot \tau \cdot \frac{\text{vol}(V_f)}{\text{vol}(V)}, \tag{A.17}$$

which by Lemma 3 is a contradiction. □

Finally, we prove an upper bound on $(f/c) \cdot \text{vol}(V_f)$, as follow.

Theorem 2: *Given a social graph with a fast mixing real region and an attacker that randomly establishes attack edges, the number of fake nodes that rank similar to or higher than real nodes after* $\beta = O(\log n)$ *iteration is* $O(\text{vol}(E_a) \cdot \log n)$.

Proof. Consider a social graph G and its defense graph D. By Lemma 1, we know that re-weighting G changes its mixing time by only a constant factor. Therefore, we have $\mathcal{T}_D(\varepsilon) = O(\mathcal{T}_G(\varepsilon))$ and $\mathcal{T}_{D_r}(\varepsilon) = O(\mathcal{T}_{G_r}(\varepsilon))$. As G_r is fast mixing, we also have $\mathcal{T}_{G_r}(\varepsilon) = O(\log n)$ by definition. This also means $\mathcal{T}_{D_r}(\varepsilon) = O(\log n)$. As $V_f \neq \emptyset$, then by the bound in Equation A.9, we have $\mathcal{T}_D(\varepsilon) - \mathcal{T}_{D_r}(\varepsilon) = \Omega(1)$. So, $\mathcal{T}_D(\varepsilon) - \beta = \Omega(1)$ as $\beta = \mathcal{T}_{D_r}(\varepsilon) = O(\log n)$. Finally, by Lemma 4, we know that at most $(f/c) \cdot \text{vol}(V_f)$ fake nodes can rank same or equal to real nodes. We now attempt to prove an upper bound on this quantity.

As the total trust τ is conversed after $\beta = O(\log n)$ iteration, we have

$$f \cdot \tau \cdot \frac{\text{vol}(V_f)}{\text{vol}(V)} + c \cdot \tau \cdot \frac{\text{vol}(V_r)}{\text{vol}(V)} = \tau \cdot$$

That is,

$$\frac{f}{c} \cdot \text{vol}(V_f) = \frac{\text{vol}(V_r)}{\frac{\text{vol}(V)}{f \cdot \text{vol}(V_f)} - 1} \cdot$$

By Lemma 3, we have

$$\frac{f}{c} \cdot \text{vol}(V_f) = \frac{\text{vol}(V_r)}{(\tau/\pi_\beta(V_f)) - 1} \cdot \tag{A.18}$$

Now, by Lemma 2 and Corollary 2, we have:

$$\begin{aligned}
\pi_\beta(V_f) &= \sum_{0 \le i \le \beta - 1} (\chi(V_r) \cdot \tau) \left(1 - \chi(V_r) - \chi(V_f)\right)^i \\
&< \sum_{0 \le i \le \beta - 1} (\chi(V_r) \cdot \tau) \left(1 - \chi(V_r)\right)^i \\
&= \sum_{0 \le i \le \beta - 1} \tau \cdot \left((1 - \chi(V_r))^i - (1 - \chi(V_r))^{i+1} \right) \\
&= \tau \cdot \left(1 - (1 - \chi(V_r))^\beta \right)
\end{aligned} \tag{A.19}$$

By combining Equations A.18 and A.19, we get:

$$\frac{f}{c} \cdot \text{vol}(V_f) < \text{vol}(V_r) \left((1 - \chi(V_r))^{-\beta} - 1 \right)$$

By replacing $(1 - \chi(V_r))^{-\beta}$ with $1 + \beta \cdot \chi(V_r) + o\left(\chi^2(V_r)\right)$, which is its Maclaurin

series, we end up with the following

$$\begin{aligned}\frac{f}{c}\cdot \mathrm{vol}(V_f) &< \mathrm{vol}(V_r)\left((1-\chi(V_r))^{-\beta}-1\right)\\ &= \mathrm{vol}(V_r)\cdot O\left(\chi(V_r)\right)\cdot\beta\\ &= \mathrm{vol}(V_r)\cdot O\left(\chi(V_r)\right)\cdot O(\log n)\\ &= O\left(\mathrm{vol}(\partial(V_r))\cdot\log n\right)\\ &= O\left(\mathrm{vol}(E_a)\cdot\log n\right),\end{aligned}$$

which completes the proof. □

Appendix B

Evaluating Sybil Node Detection Algorithms with SyPy

SyPy is a Python package for Sybil node detection in social and information networks. We designed SyPy to evaluate the effectiveness of graph-based Sybil node detection algorithms on a small scale, that is, networks which consist of thousands of nodes and can fit into a single commodity machine's memory. The package is not designed to benchmark the efficiency of these algorithms on a large scale, as there are existing distributed systems which better support large-scale evaluations.

B.1 Framework

SyPy provides an extensible framework to design, implement, and evaluate graph-based Sybil node detection algorithms. For example, SyPy includes 8 off-the-shelf algorithms for fake account detection in online social networks (OSNs), including Íntegro and SybilRank. In what follows, we describe the main abstractions defined in this framework.

B.1.1 Graphs and Regions

A *region* has a well-defined *graph* structure and is either Sybil (i.e., fake) or honest (i.e., real). For example, one can create a Sybil region consisting of 100 nodes that are fully-connected (i.e., a complete graph) as follows:

```
import sypy

sybil_region = sypy.Region(
    graph=sypy.CompleteGraph(num_nodes=100),
    name="SybilCompleteGraph",
    is_sybil=True)
```

SyPy supports many random graph models, such as scale-free and small-world graphs. For example, one can create an honest region to model a community of 5000 users as follows:

```
honest_region = sypy.Region(
    graph=sypy.SmallWorldGraph(
        num_nodes=5000,
        node_degree=100,
        rewiring_prob=0.8),
    name="HonestSmallWorldGraph")
```

In order to specify known honest accounts, one can pick nodes at random from the honest region as follows:

```
honest_region.pick_random_honest_nodes(num_nodes=10)
```

B.1.2 Networks

A *network* always consists of two regions: The honest region positioned to the left, and the Sybil region positioned to the right. The regions can have any graph structure, and initially, they are disconnected. We can "stitch" the two regions together

in any way that resembles how Sybils connect with honest nodes in real-world networks (e.g., using a random or targeted infiltration strategy). For example, one can use the regions defined above to create a network and connect ten randomly picked pairs of nodes—one node from each region in each pair—as follows:

```
social_network = sypy.Network(
    left_region=honest_region,
    right_region=sybil_region,
    name="OnlineSocialNetwork")

social_network.random_pair_stitch(num_edges=2)
```

B.1.3 Detectors

After the network is created, one can run any of the supported graph-based Sybil node *detectors* and compute their detection performance. For example, we can use SybilRank for fake account detection in OSNs as follows:

```
detector = sypy.SybilRankDetector(social_network)
results = detector.detect()

print "sensitivity={0:.2f}, specificity={1:.2f}".format(
    results.sensitivity(),
    results.specificity())
```

B.2 Benchmarks

SyPy offers a set of benchmarks to evaluate the effectiveness of detectors using ROC analysis, given a suitable operating threshold such as a pivot along a ranked list of nodes. Benchmarks also compute the curve's AUC, and are generally used to perform sensitivity analysis.

```
ranking_benchmark = sypy.MultipleDetectorsBenchmark(
    detectors=[sypy.SybilRankDetector, sypy.IntegroDetector],
    network=social_network,
    thresholds=["pivot", "pivot"])
ranking_benchmark.run()

ranking_benchmark.plot_curve(file_name="roc_analysis.pdf")

for benchmark in ranking_benchmark:
    print "detector={0}, auc={0:.2f}".format(
        benchmark.detector,
        benchmark.auc)
```

SyPy also supports advanced benchmarks that allow evaluating detectors against a dynamic network, where graph structures can change such as the establishment of new attack edges. One can benchmark detectors under different number of attack edges as follow, where the output is a curve comparing the number of attack edges to the AUC of each detector.

```
edges_benchmark = sypy.AttackEdgesDetectorsBenchmark(
    multi_benchmark=ranking_benchmark,
    values=[1,10,100,1000,10000]) # attack edges
edges_benchmark.run()

edges_benchmark.plot_curve(file_name="edges_vs_auc.pdf")
```

B.3 Extensibility

SyPy is built on top of NetworkX,[1] and hence it can support most of NetworkX functionality, which also means it is easily extensible. For example, one can visualize any region or network as follows:

[1] http://networkx.github.io

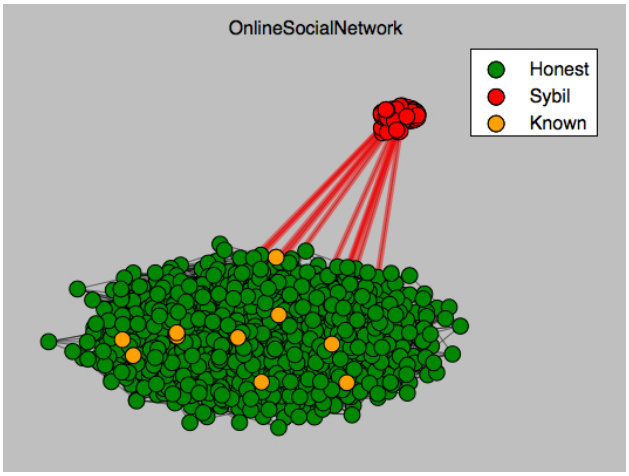


Figure B.1: SyPy network visualization

```
sybil_region.visualize()
sybil_region.visualize(file_path="{0}.pdf".format(sybil_region.name))

social_network.visualize()
social_network.visualize(file_path="{0}.pdf".format(network.name))
```

Figure B.1 depicts how a network visualization in SyPy looks like. Notice that SyPy color-codes the important elements of the network, along with a useful legend. In addition, one can zoom in and out, move the graph left or right, and save the visualization in many formats interactively.