Innocent by Association: Early Recognition of Legitimate Users

Yinglian Xie, Fang Yu, Qifa Ke, Martín Abadi[†][*]
Microsoft Research Silicon Valley

Eliot Gillum, Krish Vitaldevaria, Jason Walter
Microsoft Corporation

Junxian Huang[†], Z. Morley Mao[†]
University of Michigan

ABSTRACT

This paper presents the design and implementation of *Souche*, a system that recognizes legitimate users early in online services. This early recognition contributes to both usability and security. Souche leverages social connections established over time. Legitimate users help identify other legitimate users through an implicit vouching process, strategically controlled within vouching trees. Souche is lightweight and fully transparent to users. In our evaluation on a real dataset of several hundred million users, Souche can efficiently identify 85% of legitimate users early, while reducing the percentage of falsely admitted malicious users from 44% to 2.4%. Our evaluation further indicates that Souche is robust in the presence of compromised accounts. It is generally applicable to enhance usability and security for a wide class of online services.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; H.4.3 [**Communications Applications**]: Subjects—*Electronic mail*

Keywords

Social graph, vouching, account hijacking, legitimate user recognition

1. INTRODUCTION

In the early days of online services, users were generally presumed innocent until proven guilty. This attitude was tenable as long as large-scale attacks remained rare, and it led to pleasant, frictionless user experiences. Unfortunately, today, online services have become a popular platform for attackers to conduct a variety of malicious activities including sending spam, performing social engineering attacks, and propagating malware. These activities all rely on the use of a large number of attacker-created accounts and compromised accounts as vectors to deliver malicious contents. The relentless abuse of online services has led to a more cautious stance. It has also led to much user annoyance, directed at both the abuse and some of the defenses, which however do not guarantee security.

Most existing defense efforts focus on the *detection* of malicious accounts, often by analyzing user behaviors (e.g., logins or message postings) [34, 18]. Establishing user reputation through behaviors can often take a long time [7]. Before a new user has enough activity history, many service providers impose strict usage restrictions in order to mitigate attack damage and to prevent new attacks. Notable examples of such restrictions include aggressively requiring CAPTCHA tests every now and then (not just for account creation), and limiting the number of outgoing emails [3]. Unfortunately, such strategies are ineffective at stopping attackers who can easily outsource CAPTCHAs to third-party cheap labor [2], for example. More seriously, they often negatively impact legitimate user experience. CAPTCHAs are often hard for humans [12], resulting in both inconvenience and user frustration.

Social connectivity is another basis for defense, since a legitimate user would rarely want to establish connections with attackers. In particular, a number of Sybil defenses have been proposed to detect attacker-created identities (i.e., Sybil nodes) [32, 31, 15]. These proposals and other work on inferring trust [17] rely on graph structures for detection. However, recent measurement studies indicate that two of the main assumptions on graph structures required by Sybil defenses, i.e., fast mixing social networks and the existence of a tight Sybil community, do not hold on real social graphs [22, 30]. The existence of compromised accounts further undermines those assumptions.

Leveraging social connections in a different manner, the goal of our work is to recognize legitimate users early on, without inconveniencing them and without sacrificing service security. Our premise is that a single social connection originating from a known good user is often sufficient to endorse the legitimacy of the receiver. This endorsement is a lightweight process, which we call *vouching,* that allows us to recognize legitimate users early. Vouching is more efficient than methods that rely on entire social network structures. We show that, unlike typical invitation systems, vouching can be completely transparent to users without requiring explicit user cooperation.

However, this approach presents a number of challenges in practice. How do we define vouching? Who can serve as vouchers? How do we ensure the robustness of vouching in the presence of active adversaries? As attackers can compromise legitimate user

[*]Martín Abadi is also affiliated with the University of California, Santa Cruz.

[†]The work was done at Microsoft Research Silicon Valley.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CCS'12, October 16–18, 2012, Raleigh, North Carolina, USA.
Copyright 2012 ACM 978-1-4503-1651-4/12/10 ...$15.00.

accounts [4, 1], how do we prevent these compromised accounts from aggressively vouching for other malicious accounts?

In this paper, we present a system called *Souche* [1] that leverages social connections, established over time, to support vouching. By carefully monitoring vouching via social community structures, we show that it is possible to recognize a large subset of legitimate users as early as they start actively using a service, and at the same time to limit the growth of the malicious user population despite strong adversarial models. (Legitimate users that are not recognized by Souche will continue to be subject to the current usage restrictions.)

Souche builds upon two components. The first component constructs a social graph and selects vouchers by computing connected subgraphs. Our approach is inspired by a key observation on real data: that there exists only one giant connected subgraph of legitimate users, while malicious users are mostly isolated nodes. The second component limits the growth of the trusted user population based on community structures defined as a set of *vouching trees*. The use of vouching trees restricts the impact of active adversaries to small local subgraphs, thereby preventing the population of malicious users from growing quickly. Furthermore, it enables us to generate strong audit trails that permit reconsidering and invalidating vouching between accounts.

We implement Souche and demonstrate its effectiveness in the context of an email service where user social connections are represented by their email communications. Our analysis and experimental evaluation based on an anonymized Hotmail dataset of over 250 million sampled users shows that Souche has the following attractive properties:

- *User friendly:* Vouching is a lightweight operation, completely transparent to users. Service providers can apply vouching either in batch mode or in real time as user requests arrive, without requiring explicit user cooperation.
- *Effective in recognizing legitimate users:* Souche can identify a vast majority (85%) of the legitimate users. Among them, Souche can identify 87% as early as their first day of sending emails.
- *Effective in denying admission of malicious users:* Souche can reduce the percentage of malicious users (w.r.t. the total number of users) admitted into the system from 44% today (by CAPTCHAs) to only 2.4%, an order of magnitude reduction.
- *Robust to attacks:* Our tree-based vouching algorithm is robust to account-hijacking attacks. Even in the presence of aggressive vouching behavior from malicious accounts, 99.6% of legitimate users can still be admitted. The use of vouching trees effectively bounds the increase of the malicious user population.

Our research is an effort to return to a situation where most legitimate users can enjoy online services with few speed-bumps, from the moment that they join those services. We demonstrate the usefulness of Souche for large-scale email services. It can also be applied to a wide set of applications where we can derive social connectivity, e.g., via messenger communications, tweet mutual mentioning, or Facebook interactions.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents the system overview, followed with details in Section 4 and 5. Section 6 evaluates our approach. Section 7 discusses attacker counter-strategies and future work. Section 8 concludes.

2. CONTEXT AND RELATED WORK

In this section, we review related work and the attacks we consider. We also discuss the differences between our work and the previous Sybil defense solutions.

Despite many defense mechanisms, malicious account creation continues to be an unsolved problem. Meanwhile, attackers have also started hijacking user accounts at a large scale, making defense even more challenging. Account-hijacking is not completely new, but is increasingly common in the past two to three years [1, 4, 5]. There exist many channels for attackers to compromise a user account, including phishing, password cracking, keyloggers, viruses, and malware [5]. Our goal is to develop a solution that could increase service usability despite the existence of both compromised accounts and attacker-created accounts.

2.1 CAPTCHAs

Reverse Turing tests, including CAPTCHAs, have been ubiquitously deployed to protect open Web services from being abused by automated efforts. Although CAPTCHAs remain effective in differentiating humans from bots, the emergence of the human-solving market has largely defeated their primary purpose of defending against attackers [2, 6]. Motoyama et al. [23] have performed a large-scale study on the economic model of CAPTCHA-solving services. They show that the price of CAPTCHA-solving continues to decline, further diminishing their effectiveness for defense.

2.2 Why not Sybil Defense Solutions?

The value of using social networks to detect fake identities has been widely studied in the context of P2P and other decentralized systems. A number of solutions have been proposed to detect or prevent Sybil attacks [16, 32, 31, 15, 26, 24, 25]. However, we cannot directly adopt these techniques for a number of reasons.

First, existing Sybil defenses are generally designed for decentralized environments, where each user would like to independently determine the legitimacy of a small set of other users. In such a setting, the knowledge of an existing good user who wishes to perform Sybil tests is a given. On the other hand in our scenario, we would like to identify new legitimate users based on a collection of unclassified users.

Second, Sybil defenses and other trust inference work [17] rely on social graph structures for detection [27]. These solutions are often heavyweight, e.g., requiring random walks that traverse a large number of nodes for each test. They also rely on strong assumptions regarding social graph topologies. However, recent measurement studies show that two key assumptions required by Sybil defense algorithms may not hold on real social graphs: these social graphs are not as fast mixing as required by the existing solutions [22], and real Sybil nodes scatter around the entire graph instead of forming their own separate communities [30]. In addition, the existence of many compromised (i.e., hijacked) accounts may further invalidate the assumptions of Sybil work, where it is expected that the legitimate user region and the Sybil region are well separated with a few connections between them. Instead, each compromised account could connect to a large number of malicious accounts, making the two regions highly intertwined.

In contrast, our goal is to identify legitimate users early. We make no assumptions on the social network topologies. Rather than inferring trust from graph structures as in previous Sybil defense

[1] In French, "souche" means tree stump, but also origin or source, by analogy (often with positive connotations, sometimes with xenophobic intent). The association with trees and with lineage seems appropriate for our system, as does the lexical similarity with "e-social-vouching".

or trust inference work, we leverage lightweight vouching that reflects direct trust from known good users. We guard the growth of malicious user population using tree-based community structures to tolerate the existence of both attacker-created and compromised accounts. By controlling vouching-tree sizes, we can thus effectively bound the number of admitted malicious users. For legitimate users, while we do not provide formal guarantees, we show that a majority of them can be admitted using empirical evaluations on large datasets. Further, vouching is a low-overhead operation in practice and our system is highly efficient, as shown in Section 6. Finally, both the scale of the graphs and the scale of the attacks we study are much larger than in most previous research.

2.3 Other Related Work

Social features have also been leveraged to defend against spamming and other social network attacks. For example, the use of clustering coefficients [11] and page rank [14, 13] can help classify good vs. malicious accounts. More broadly, there exist extensive studies to understand the graph properties of large online social networks including Facebook and Twitter (e.g., [20, 8, 9, 28]). Among those, the work by Zhao *et al.* [28] shows that normal social links are not an accurate representation of meaningful peer connectivity on social networks, but instead the interaction graph should be considered. This observation is helpful in picking activities for constructing social graphs.

Although our work does not focus on attack detection, it is complementary to previous approaches for detecting spamming attacks by email contents, network-level features, or suspicious login activities (e.g., [34, 29, 19]). In particular, it can be used in combination with the user reputation systems deployed by many online services (e.g., [7]). For example, Souche can pick users with established good reputations to vouch for other new legitimate users early. If an already endorsed account was assigned a low reputation score later (e.g., after being hijacked), this information can be used to prevent the account from further vouching for others and to detect the existence of other malicious accounts based on vouching causalities (see Section 5.5).

3. SOUCHE OVERVIEW

This section presents the high-level overview of our system design. We elaborate on its details in the next two sections.

3.1 Our Goal

When a user creates an account with an online service, the user will typically receive a CAPTCHA to solve. However, after the user successfully solves the CAPTCHA and registers an account, the user will often continue receiving CAPTCHAs at an aggressive rate (e.g., after sending every ten emails) and being subject to usage limitations until the user establishes a good reputation.

Souche does not aim to eliminate the CAPTCHA tests at the account signup time. The primary purpose of Souche is to recognize *new legitimate* users earlier, *after* they have created accounts, but *before* they establish long legitimate usage histories, so that we can remove unnecessary CAPTCHAs or other service restrictions for new users. Users who are recognized as legitimate by Souche are considered trustworthy and can benefit from relaxed service usage policies. In contrast, those who are not identified by Souche will be subject to the current practice (CAPTCHAs, various usage limitations, etc.), until they establish reputation gradually. While false negatives (legitimate users not identified by Souche) are undesirable, these users may be treated no worse than they are today.

As we focus on early recognition of legitimate new users, Souche is not an attack detection system in that unidentified users are not always malicious. However, the new user accounts recognized by Souche may later be compromised by attackers. While Souche does not address attack detection, one of our design goals is to ensure that the system is robust in the presence of both compromised accounts and attacker-created accounts and to restrict the growth of the malicious account population.

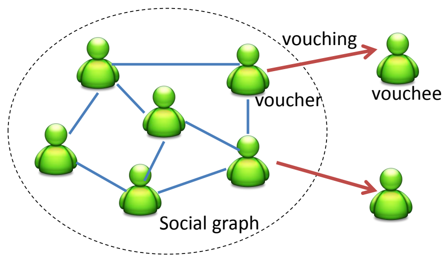


Figure 1: Vouching from trusted users on the social graph.

3.2 Vouching

Souche leverages social connectivity for determining user legitimacy in a lightweight manner. Its core idea is to let good users help identify other good users through a *vouching* process, as illustrated in Figure 1. Once being vouched for, a user (called *vouchee*) can in turn vouch for others and become a *voucher*, and so on. If a vouchee is later seen to be either a malicious or a compromised user, it will be *devouched* and will not be able to vouch for any new users subsequently.

Vouching activity is thus directional and reflects direct trust. It does not necessarily need to be an explicit action, but can be implicit based on many different forms of user activities. For example, in an email-service application, vouching activity can be naturally defined as sending emails. In online social network or instant messaging applications, vouching activity can take the form of extending invitations, making recommendations, or tweet following. In file-sharing applications, vouching activity can be defined based on the data-sharing relationship between users, e.g., where user A sharing data with user B implies A vouching for B.

Finally, the *vouching process* is performed by service providers at the back end and can be fully transparent to users. Service providers can either batch all vouching activities to process periodically (e.g., on an hourly or daily basis), or process each activity as it happens in real time to identify legitimate users instantly. In addition, service providers can strategically choose the subset of activities that qualify as vouching.

3.3 Security vs. Usability

Souche focuses on identifying and protecting legitimate users. With significantly improved ability to recognize a large subset of good users (especially active ones), service providers may further increase the frequency of CAPTCHAs or enforcing other authenti-

	new good	new bad	old good	old bad (malicious or compromised)	
				detected	undetected
w/o Souche	✓	✓	✗	removed	abuse
w/ Souche	✗	✓✓	✗	removed	abuse + vouch

Table 1: Security and usability models with and without Souche. ✓means CAPTCHAs and service restrictions;✗means otherwise. ✓✓means more aggressive CAPTCHAs and restrictions. "Removed" means the account is removed from the system.

cation channels (e.g., using SMS) for the remaining set of users, so as to raise the cost significantly for attackers.

Table 1 summarizes the usability and security models with and without Souche, depending on the user category. We define a newly registered user as a *new user*. Without Souche, a new (good or bad) user will receive CAPTCHAs and other service limitations before the user establishes reputation, at which point the new user becomes an *old user*. With Souche, our goal is to avoid imposing CAPTCHAs or service restrictions on new good users, while still applying these measures to new bad users. For old, undetected bad users (compromised or malicious accounts, but with good reputations), without Souche, they will send spam or perform other malicious activities to abuse the service. With Souche, these old, undetected bad users may additionally vouch for new malicious users. However, Souche is designed to restrict the number of new malicious users that will be vouched for to be in proportion to the population size of the undetected compromised users. We assume compromised users are usually a small fraction of the entire user population. In the extreme case where attackers successfully compromise a large percentage of good users, Souche will not be effective. However, at that point, the service will become unusable and we can only resort to attack detection and recovery systems.

For services with higher security requirements and less usability concerns, Souche can be used in combination with CAPTCHAs to enhance the security of the existing CAPTCHA-based defense solutions. Even if attackers can solve CAPTCHAs with costs, it is still difficult for them to establish vouching relations from trustworthy accounts to malicious accounts.

3.4 Challenges and Design Overview

In the ideal case, vouching reflects trust from one legitimate user to another. In practice, there are two major challenges in the design of Souche. The first is the selection of vouchers. How should we pick vouchers to bootstrap the system? Over time, how does the set of vouchers grow? The second challenge is to limit the number of malicious accounts being admitted. Given that attackers are constantly creating user accounts and compromising existing accounts, how do we prevent attackers from polluting the voucher set?

To support vouching, Souche builds upon two components. The first component bootstraps the system by constructing a social graph and leveraging graph properties to identify a set of trusted users to serve as the initial set of vouchers. We analyze two real, large datasets to motivate our design choices. Our study helps us select existing good users as vouchers by computing connected social graph components. This approach is general and independent of the specific user activities in different applications. The second component determines when and whether vouching is allowed. We explicitly build community structures in the form of vouching trees so that we can monitor and control the growth of the voucher population. In the presence of active adversaries, the use of well separated communities restricts their damage and growth to small local subgraphs, thereby preventing the population of malicious users from growing quickly or polluting the voucher set globally.

3.5 Datasets

We have access to the following two datasets, and primarily focus on the email dataset for our study since it is larger and has user reputation scores for evaluation. We also use the Twitter dataset to study the generality of important graph properties that we explore.

Email service dataset: We use an anonymized dataset containing coarse-grained email communication information on 269.7 million uniformly sampled user accounts (referred to as *Sampled-users*, see Table 2), collected during a 900-day period from October 2007 to April 2010 at Hotmail. Each entry contains a user ID hash and its communication history summary with another user ID hash, including the number of emails the two users have exchanged and the first and the last sending timestamps.

We additionally have access to a reputation dataset from Hotmail, containing a superset of 660 million sampled users, referred to as *All-users* (Table 2). Each user is associated with a registration date and a reputation score derived from user sending history to help detect spamming accounts. The reputation scores are not completely accurate, but can help sanity-check the legitimacy of an account *after* the account has been active for a while.

We use the term *internal users* to refer to the user accounts from Hotmail. Since the communication records of the sampled internal users may also involve users from other email services, we refer to the users from other services as *external users*. Both the set of All-users and the set of Sampled-users consist of internal users.

Twitter dataset: We collect one month of public tweets, in total 1,475,522,405 tweets, from August 2011. The Twitter community follows a few stylistic conventions when composing tweets, one of them the convention of mentioning other Twitter users by prefixing their Twitter user IDs with the @ character. We use this convention to extract user IDs for constructing a mutual-mention graph.

4. BOOTSTRAPPING VOUCHING

In this section, we present how to bootstrap Souche with an initial set of vouchers. For new services with a small number of users, one may afford to pick known good users manually. For large services with existing user populations, one may leverage existing user reputations or attack detection systems to select initial vouchers. These detection systems often rely on application-specific user activities for establishing user reputation, so may not be effective at labeling new users. Nevertheless they could help classify old users after they have long enough usage histories.

We use a more general method that leverages the social graph structure (combined with user reputation systems if they already exist) to help select initial vouchers. We take a data-driven approach and study distinguishing graph characteristics that result from collective user behaviors on two different datasets.

4.1 Social Graph Construction

We use social graphs to represent the connectivity among users based on their communications or activities that reflect their social relationships. Our social graphs are more general than the common friendship graphs such as those explicitly built in Facebook or Google+.

To prevent malicious accounts from connecting to a large number of legitimate users, we require strong mutual user connections that are less likely to be gamed by attackers. For example, for the Twitter application, if two users have mutually mentioned each other in their tweets, it implies that a social bond does indeed exist and it is hard for attackers to spoof such connections with legitimate users. Thus we use an undirected graph $G_u = (V_u, E_u)$ to model a social graph, where a node represents a user and an edge represents mutual user actions (e.g., friendship invitation and confirmation, mutual email exchanges).

Different applications could explore application-specific semantics for graph construction. For instance, instant messaging applications may construct graphs based on frequent mutual messages. Web forums could construct user mutual post-reply graphs. Even in social network applications such as Facebook and Google+, it may be more desirable to use the actual user interaction graph (e.g., who commented on whose page) than using the default friendship graphs [10, 28]. The reason is that many users may not be prudent in confirming friendship requests, and may unintentionally establish friendship connections with sophisticated malicious users.

4.2 Email and Twitter Graphs

In the email application, we construct the social graph based on mutual email exchanges. We consider only user pairs in which each user has sent at least two emails to the other one in the pair (i.e., ≥ 2 emails from A to B, and ≥ 2 emails from B to A) to preclude weak connections because of unintended, accidental events (e.g., clicking the "reply" button to an unsolicited email). We remove all users who have sent to a large number (≥ 5000, a threshold set empirically) of unique recipients, as they likely correspond to newsletters or spammers rather than normal users. Using the communication records of the 269.7 million Sampled-users, we obtain an *email-friendship graph* with 255,096,776 nodes (with both internal and external users) and 436,872,175 undirected edges, with an average degree about 3.425. Note that not all Sampled-users are on the graph due to our graph construction requirements. Two factors might explain the relatively small degree. First, the email friendship graph is an interaction graph and requires two email mutual exchanges to have an edge. Second, the graph is built from a sampled dataset, where we do not have access to the email communications among external users.

Using the Twitter dataset, we construct the *tweet-mention graph* derived from tweets by extracting user pairs that have mutually mentioned each other at least once. The corresponding graph contains 10,410,144 nodes and 176,551,621 undirected edges from 1,265,660,845 related tweets.

4.3 Giant Connected Component

The small-world phenomenon and previous work both suggest that there exists a giant connected component (GCC) on social graphs [21]. The questions are "to what degree is this hypothesis true?" and "how can we leverage this observation?"

All-users (internal)	Sampled-users (internal)	GCC-users (internal, external)	GCC-I users (internal)
660M	269.7M	237.3M	73.5M

Table 2: Categories of users derived from the email datasets.

We compute all the connected components on both the email-friendship graph and the tweet-mention graph. On both graphs, there indeed exists a GCC that includes the majority of users. For the email-friendship graph, we find that the GCC contains 237.3 million nodes, accounting for 93% of nodes on the constructed graph. We refer to the particular set of 237.3 million users as *GCC-users*; they include both internal and external users. For the tweet-mention graph, the GCC has 9.17 million nodes (88% of all nodes). We find that the remaining connected components are all *very* small. The second largest connected component on the email-friendship graph has only 283 nodes, and on the tweet-mention graph, it has only 556 nodes. Their sizes are orders of magnitude smaller than those of the giant connected components.

Next, we are interested in where good and malicious users are located on the graphs. We use the reputation dataset of All-users from the email service to examine user legitimacy. (We do not have reputation scores for Twitter users.) Among 237.3M GCC-users, 73.5 million are internal users for whom we have reputation scores (referred to as *GCC-I users*). Table 2 lists the sizes of these different user populations. Figure 2 shows the reputation distribution into four categories from good to bad: good, marginal, suspicious, and malicious. Because of a lack of history, new users will by default be classified as marginal. Compared with All-users, we find that

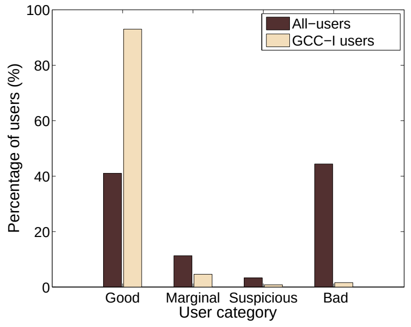


Figure 2: The reputation score distribution of GCC-I users

GCC-I users are predominantly (93.0%) good users. Only 1.6% of GCC-I users are classified as bad, as opposed to 44.4% of All-users.

Summary of key findings: Our study thus not only confirms the existence of a big "island" of a majority of users, but also generates the following findings:

- *There exists only one giant connected component (GCC) of users on a social graph with strong mutual connections; the remaining connected component sizes are orders of magnitude smaller.*
- *GCC users are predominantly good users. Very few malicious nodes are on the GCC; they are more likely isolated nodes.*

Our study indicates that computing GCCs on social graphs is a practical way for identifying a large set of legitimate users as vouchers. If account-hijacking is a concern, service providers may select historical snapshots of graphs without many compromised users. In addition, they could also combine the knowledge of a GCC with existing systems for user reputation or attack detection to further remove already identified malicious users.

5. THE VOUCHING ALGORITHM

While GCC users provides a starting point for voucher selection, relying on computing the GCC periodically alone for identifying all legitimate users has two problems. The first problem is the delay for recognizing new users, as strong mutual social connections may be relatively slow to establish. The second, more important issue is the robustness to compromised and attacker-created accounts. There may exist on the GCC a very small number of malicious users from the beginning. Attackers may also wish to pollute the set of GCC users by compromising a subset of them over time, and let each of them connect with many attacker-created accounts.

To address the above problems, Souche monitors the graph dynamics and controls the growth of the vouched user population over time. In particular, we rate-limit the number of new vouchees based on community structures represented as a set of *vouching trees*. Therefore, even in the presence of compromised or malicious vouchers, we can effectively restrict the growth of the malicious user population within their individual small communities.

The growth rate usually depends on the population growth model and the application. A service provider can pick a growth rate based on its resource provisioning, the historical rate, and its specific growth phase (e.g., an initial booming phase or a stable phase). The selected rate does not need to reflect the true population growth demand accurately. In particular, applications can determine the specific rate to control vouching coverage vs. security. A more conservative rate does imply that fewer users will be vouched for.

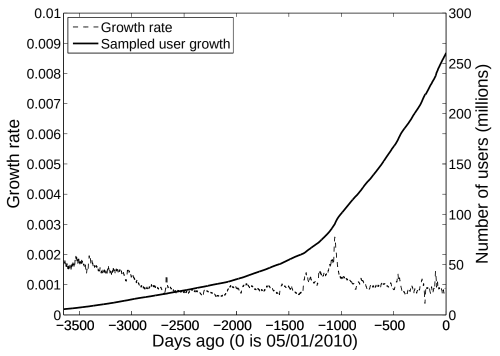


Figure 3: Good user population growth and its growth rate over a 10 year period. The left Y-axis shows the daily growth rate. The right Y-axis shows the population size.

But since the remaining users will fall back to the existing service restrictions (e.g., receiving CAPTCHAs), they will be treated as they are today.

5.1 Population Growth Rate Selection

We analyze the email reputation dataset to study the population growth. The purpose is not to generalize the particular growth rate, but to provide an example normal-user growth model for a real, large service, and to show that it is feasible and practical to derive a growth rate for a specific time period based on history.

We extract all users with good or marginal reputation scores and plot the user population growth (using the registration dates) for the past 10 years until April 30, 2010 (marked as Day 0 on the X-axis) in Figure 3. We also plot the daily growth rate of the user population (smoothed using a 10-day sliding window) on the same graph. We observe from the figure that while the user population grows exponentially over time, the *growth rate* is relatively stable over time, between 0.001 to 0.002.

The growth of the good user population has a stable rate, which can be used to throttle vouching periodically, for example, on a daily basis. With a fixed rate, however, simple throttling schemes have many problems. Below, we discuss two straightforward solutions and their pros and cons. We then present the algorithm we propose and its security properties.

5.2 Strawman Solution I: Global Quota

With a prior selected growth rate r, one straightforward solution is to compute a global quota in terms of the total number of new vouchees that are expected to be admitted into the system periodically. Let us assume that time is a discrete sequence $t_0, t_1, t_2, ...$ (for example, t_i represents day i).

We use n_i to represent the number of vouchers present at time t_i, so n_0 is the number of vouchers at the beginning. We use Q_i to denote the global quota at time t_i and we allow quota to be fractional. Then Q_i can be computed as:

$$Q_i = n_{i-1} \times r$$

We allow all vouching operations to succeed until the number of new vouchees reaches the quota Q_i for t_i. Any subsequent vouching will then be queued, until the system has new quota at the next time t_{i+1}. So $n_i \leq n_{i-1} + Q_i$.

Pros: The global quota system is simple to implement and ensures that the total number of vouchers increases according to a specified global rate.

Cons: Even when a small number of vouchers are compromised, attackers can aggressively vouch for new malicious accounts and greedily use up all the quota in the system at every time step, denying quota for legitimate users.

5.3 Strawman Solution II: Local Quota

We could also compute a local quota for each voucher v_j and independently determine whether v_j can vouch for another user. With a fixed growth rate r, for a single user v_j who gets admitted into the system at time b_j, the maximum number of total users for whom v_j could have vouched at time t will be $(1+r)^{t-b_j} - 1$. If v_j has already vouched for c_j users (i.e., c_j vouchees) by t, then the remaining number of users v_j can vouch at t, defined as the local quota q_j for v_j at t, can be computed as:

$$q_j = (1 + r)^{t-b_j} - c_j - 1$$

Pros: The local quota system is also simple to implement. Since the growth rate of each node is at most r, it ensures the global growth rate will be no more than r. Further, the scheme is more robust to account-hijacking attacks and ensures the growth rate of malicious accounts to be no more than r as well.

Cons: Although local quota enables fair sharing of vouching quota among users, it is too restrictive in admitting new users. In practice, the number of social connections among users often has a wide variation, which cannot be accommodated by the local quota scheme. Further, given that the growth rate r is typically small, e.g., 0.001 or 0.002 on a daily basis, a new legitimate voucher will not be able to vouch for any user during almost its first year. On the other hand, many old legitimate vouchers may be inactive in vouching for new users, hence they are unable to fully utilize their quota.

5.4 Our Solution: Tree-based Quota

The global and local quota schemes are the two extremes of the solution space. The local quota scheme isolates quota and is secure, but not practical to use. In contrast, the global quota scheme enables quota sharing, but not secure.

To address this dilemma, our solution includes two key ideas. First, we allow quota sharing, but within only a subset of users in order to enforce quota isolation. Second, since vouching reflects one type of social connectivity, we leverage the vouching relations among users to build tree-based communities for quota sharing and isolation. Specifically we use a set of *vouching trees* to represent the user vouching relationship. On each tree, a parent node is the voucher of its children nodes, with the root being an initial voucher. Figure 4 shows an example.

With vouching trees, nodes can share quota with a selected subset of other nodes on the same tree, but nodes on different trees cannot share quota with each other. Intuitively, when a node who wishes to vouch for a new user does not have enough quota (e.g., a new node A in Figure 4), instead of examining the node in isolation, we could determine whether to allow vouching based on its surrounding nodes on the vouching tree. If the parent of A is reputable with additional quota, or A has reputable siblings with few vouchees, then A is also likely legitimate. So we may allow A to vouch by borrowing quota from others. On the other hand, if A's parent or siblings have aggressively used up their quota, then A is also suspicious and should not be allowed to vouch.

Finally, we limit the vouching tree sizes. The use of vouching trees supports both quota sharing and quota isolation. We describe our detailed vouching algorithm in Section 5.4.1, the security properties in Section 5.4.2, and how to limit tree sizes in Section 5.4.3.

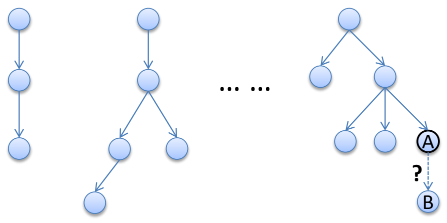


Figure 4: Example vouching trees. A is a leaf node user and wishes to vouch for a new user B.

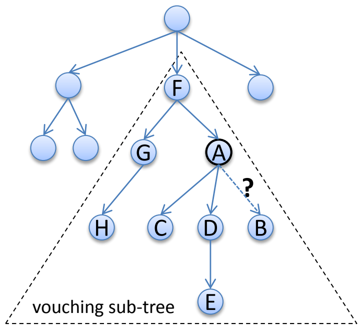


Figure 5: An example vouching subtree.

Note that a vouching tree is one way to represent a community. An alternative way is to use a subgraph (e.g., all email communication neighbors). We favor a tree structure because it is simpler, so we understand its security properties better. Exploring graph structures for different vouching strategies may be an interesting direction of future work.

5.4.1 The Algorithm

To support quota sharing on a vouching tree, each node v_j additionally maintains a quota debit d_j to keep track of how much quota v_j has lent to other nodes and hence needs to be deducted from its own local quota. With debit, a node v_j's quota at time t will be computed as

$$q_j = (1+r)^{t-b_j} - c'_j - 1 - d_j$$

where c'_j is the number of vouchees that have been vouched for by v_j using v_j's own quota without borrowing from others. When v_j borrows quota to vouch for a new user, the set of nodes who lent quota to v_j will have their debits increased to maintain the tree quota balance. Thus, the quota q_j for v_j should not be further reduced in such cases.

We use the example in Figure 5 to describe the algorithm for determining whether a user A is allowed to vouch for B at time t in the following two steps:

Step 1: quota search. In this step, A searches for available quota for vouching. The process begins with A computing its own local quota. If A has enough local quota q_A to vouch (i.e., $q_A \geq 1$), then vouching is allowed. Otherwise, A seeks to borrow quota from nearby nodes by computing the sum of the quotas from the *vouching subtree* T rooted at A's parent (illustrated by the dashed triangle in Figure 5). The set of nodes on the vouching subtree is the only set of nodes that can share quota with A, including A's descendants, node F who vouched for A, A's siblings and their subtrees. If the sum of the quotas on T (including A's local quota q_A) is greater than 1, then vouching is allowed.

Step 2: quota deduction. Without quota borrowing, $q_A \geq 1$, so quota deduction is straightforward. With quota borrowing, $q_A < 1$, all the nodes on A's vouching subtree will be affected equally by incrementing each of their debit by $\frac{max(1-|q_A|,1)}{m-1}$ (since the amount of borrowed quota won't be more than 1), assuming the vouching tree has in total m nodes, including A.

5.4.2 Guarantees

This basic tree-based algorithm has the following two properties. (See Appendix A.)

User population growth property: If the growth rate is a constant r, then the global user growth rate across all vouching trees will be no more than r.

Malicious-user population growth property: In a simplified scenario where trees are of the same height and every node behaves identically, i.e., has the same out-degree and vouches for new nodes with rate r exactly, then with α fraction of nodes being compromised and uniformly distributed in the system, the percentage of admitted nodes that are malicious will be no more than $(k-1)\alpha$ at any time, where k is the depth of the trees at the time when the malicious nodes first start vouching.

This property indicates that the fraction of quota that may be obtained by malicious nodes is tightly related to the vouching tree depth, which essentially determines the tree sizes in the simplified scenario. More generally, when nodes have diversified behavior, with varying degrees and vouchees, the vouching tree size is the important factor impacting the degree of quota sharing. In the extreme case of one global tree for all nodes, malicious nodes may use up all the available quota. Smaller trees provide better quota isolation and are thus more robust to attacks. One way to enforce tighter quota bounds for malicious nodes is to split big vouching trees into multiple small ones.

5.4.3 Optimizing Quota Search

In practice, users often have diverse behavior, so the vouching trees may have different sizes. To limit tree sizes and penalize aggressive borrowing of quota, e.g., from compromised accounts, we use three heuristics to optimize quota search:

- *Small-tree heuristic:* We periodically split large trees to limit the scope of quota search. The tree splitting is a recursive process starting from the leaf nodes. Whenever we encounter a node with its subtree size more than a threshold (set to 50 in our experiment), we split the node and its subtree from the remainder of the tree.
- *Old-node heuristic:* When a node A needs to borrow quota from others, we generate a *quota path* that records the set of nodes that share quota with A. The sum of the quota on the quota path will be scaled by the fraction of old nodes to total nodes on the quota path.
- *Vouching-delay heuristic:* If A borrows quota to vouch, then A cannot borrow quota immediately again. It has to wait for at least a pre-set damping period Δt. Similarly, a new node cannot vouch immediately after it joins and needs to wait for Δt. Note that after the pre-set delay timer expires, the quota may no longer be available because of other requests that occurred during the wait period.

These three heuristics work together to limit aggressive quota borrowing. The *vouching-delay heuristic* rate limits quota borrowing, and the *old-node heuristic* disallows quota borrowing completely if a subtree is growing too quickly with many new nodes. Finally, the *small-tree heuristic* limits the range of borrowing and ensures the total number of nodes who can share quota with each other is relatively small. We evaluate the vouching robustness to attacks with and without these heuristics in Appendix B

5.5 Devouch and Traceback

The ultimate purpose of attackers compromising and creating accounts is to perform malicious activities such as spamming and malware propagation. The focus of this paper is not on attack detection, but Souche can work with existing attack detection systems (e.g., [34]). With attack detection, a subset of these attacker-created or compromised accounts will be captured over time, so removed from the voucher sets to prevent further damage.

In addition, the use of vouching trees enables us to track back and identify the parents and ancestors of the malicious accounts on the trees, so that we can potentially detect many more attacker-created or compromised nodes. For example, in Figure 5, when we detect node A as a malicious user, then the entire subtree rooted at A may be malicious. Further, A's parent F and the sibling G may also be suspicious. Thus Souche enables us to generate strong audit trails on the causality of the accounts to improve the system robustness.

6. EVALUATION

We implement Souche using Dryad/DryadLINQ [33] on top of a 200-machine computer cluster. Our system takes the raw data of the user email communication summary as input, constructs a social graph, and computes connected components. The identified GCC-users are then used to bootstrap vouching. With the use of vouching trees, vouching can be performed in parallel across multiple machines, with each data partition maintaining a subset of trees. Parallelizing vouching improves system efficiency significantly, since each vouch operation involves accessing the states of a set of nodes up and down the tree. In our experiment, the total time to process 1 million vouch events in a batch DryadLINQ job is below 5 minutes, so it is completely practical to support even near real-time vouching for most service providers today.

We evaluate Souche in two respects using the email dataset described in Section 3.5. (We do not have the reputation scores for Twitter data for evaluation.) First, we use the dataset to evaluate the feasibility of vouching. We are interested in whether users vouched for by GCC-users are indeed legitimate users, how many legitimate users we can recognize, and how fast a new user can be vouched for after joining a service. Second, we use trace-based simulation to evaluate the coverage and the security properties of the tree-based vouching algorithm: whether the vouching trees provide enough quota to legitimate users, and the robustness of the algorithm in the presence of compromised users.

6.1 Vouching Feasibility

We first study the feasibility of applying vouching to identify legitimate users in the ideal case with *no or very few* attacks. We do not apply any quota limitations in this ideal scenario. We define vouching from user A to user B as A sending at least one email to B. We take the GCC as described in Section 4 and identify the set of users who received emails from GCC-users but are *not* GCC-users themselves. We call these users *GCC-vouchees*. We then use the reputation dataset to study the reputation score breakdown for GCC-vouchees, which include only the (vouched) internal users.

6.1.1 *How Many Users Can Benefit From Vouching?*

We first examine the entire set of GCC-vouchees and their reputation distribution across the four categories from good to bad, as summarized in Table 3. (Good and marginal categories have positive reputation. New and inactive users by default have marginal reputation because of lack of history.) Intuitively, the more GCC users vouch for the same recipient, the more likely the recipient is a legitimate user. So we further classify GCC-vouchees according to their number of unique GCC senders to study whether vouching from multiple users is more reliable.

	Internal user	Good	Marginal	Suspicious	Bad
All-users	660M	41.0%	11.3%	3.3%	44.4%
Sampled-users	269.7M	65.5%	8.5%	2.0%	24.0%
GCC-I users	73.5M	93.0%	4.6%	0.8%	1.6%
1-sender	95.4M	86.0%	11.0%	0.6%	2.4%
2-sender	50.3M	88.1%	9.1%	0.6%	2.2%
3-sender	31.4M	89.1%	8.2%	0.5%	2.1%

Table 3: Reputation distribution of GCC-vouchees. For comparison, we also include the reputation breakdowns for All-users, Sampled-users, and GCC-I users.

	Tested users	GCC+GCC-vouchees	Recall
Good	176.6M	150.4M	85.2%
Good+marginal	199.6M	164.3M	82.3%

Table 4: Recall of identifying legitimate users.

As a comparison, we also show the reputation breakdown for All-users, Sampled-users, and GCC-I users, as listed in Table 2. The statistics for All-users reflect the reputation of users already admitted through existing CAPTCHAs. Compared with All-users, the Sampled-users in general have slightly better reputations as very new users (many of them malicious accounts) are less likely to be sampled to appear in the email communication dataset.

Table 3 shows that regardless of the number of GCC-senders, the majority of GCC-vouchees are likely legitimate with positive reputation. The percentage of vouched users with bad reputation is very small. (There may also be a small fraction of hijacked users.) Compared with CAPTCHA, using Souche in addition can effectively identify legitimate users by reducing the percentage of malicious users significantly to 2.4% (1-sender vouchees) from 44% for All-users and 24% for Sampled-users—an order of magnitude reduction.

We further examine our recall at identifying legitimate users. We take the Sampled-users and compute among them the set of users with good or marginal reputation as our base. These are the set of all possible good users we target to identify. Table 4 shows that among these users with good or marginal reputation (269.7M×(65.5%+8.5%)=199.6M), Souche can identify 82.3% of them by combining GCC-I users and GCC-vouchees. If we consider only good-reputation users (269.7M×65.5%=176.6M), the coverage is even higher at 85.2%. Thus Souche can potentially recognize most of the legitimate users.

Finally, Table 3 shows that users who received emails from multiple GCC senders (2-sender and 3-sender cases) are only slightly better in reputation, yet the coverage is much smaller (x-sender vouchees are a subset of $x-1$-sender vouchees, where $x = 2, 3$). This trend suggests that receiving emails from even one GCC user is already a good indication of a user's legitimacy for vouching to be effective in the email application context.

6.1.2 *What is the Vouching Delay for New Users?*

We now study how fast a legitimate new user can be vouched for after joining the service. Since "being vouched for" matters only when a user starts to actively use the service, we use two metrics for our evaluation here:

- Number of days during which a new user sends emails before the user is vouched for.

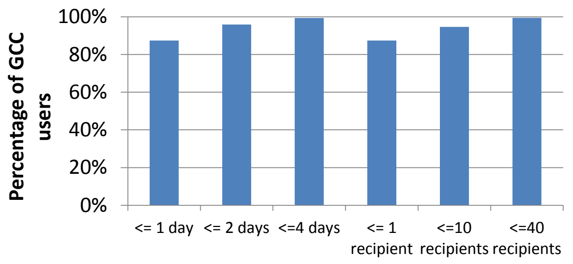


Figure 6: Vouching delay in terms of the number of active email-sending days and the number of email recipients before being vouched for by another GCC user.

- Number of email recipients a new user sends to before the user is vouched for.

From Figure. 6, we observe that 87% of GCC-vouchees have received emails from other GCC senders on their first day of email-sending. Further, 95% of GCC-vouchees have sent to no more than 10 email recipients before they receive emails from other GCC-users. Thus, for a large portion of users, vouching happens naturally as they start actively using the service. Most of them will potentially be vouched for on their first day of email activity.

6.2 Effectiveness of the Vouching Algorithm

The previous section shows the results in the case where we allow GCC users to vouch for as many users as they wish based on real data. It is thus the best case scenario where attackers do not leverage compromised accounts for vouching for malicious accounts. In this section, we include active adversaries that compromise user accounts and evaluate the robustness of the tree-based vouching algorithm using trace-driven simulations.

6.2.1 Simulation Setup

We use our email dataset to derive a sequence of events based on legitimate users joining the service and communicating with each other. Specifically, we pick GCC-users computed in Section 4.3 as our legitimate user set. We use the first-sending timestamps (accurate to the seconds) to derive events representing who is talking to whom at what time between GCC-users. We then order the events based on time and feed them into our implementation to simulate the process of vouching over time.

Given that we have only 900-day data, we do not have a complete view of user communication history from the very beginning. We select the subset of GCC-users that are active during the first year of our data collection (from October 2007 to September 2008) for bootstrapping Souche. In total, we identified 90 million users (out of 237.3 million GCC-users) as the initial set of vouchers. In addition, we identified all the vouching relationships among them to have an initial set of vouching trees. After bootstrapping, our simulation runs using the set of 334 million events derived from the data pertaining to October 2008 to April 2010.

We compare the vouching effectiveness of the tree-based algorithm against the global quota and the local quota schemes. Once a user is vouched for, the user can start vouching for others. For all three algorithms, we pick $r = 0.002$ as the population growth rate and keep it constant, and for consistency, a new node cannot vouch immediately and needs to wait for $\Delta t = 14$ days before its first vouching. For the global and the local quota schemes, to increase the robustness to attacks, we limit the total number of vouchees per voucher to be at most 50, so that no single user can aggressively vouch for many malicious users. We consider scenarios both with and without attacks.

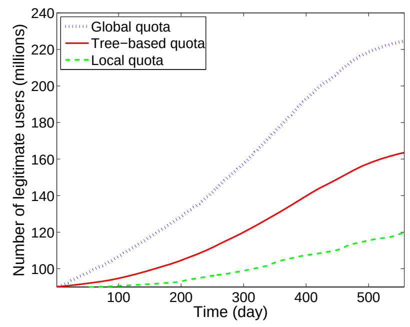


Figure 7: Number of legitimate users growing in the system over time without attacks. We start with 90 million users.

6.2.2 Coverage of Legitimate Users

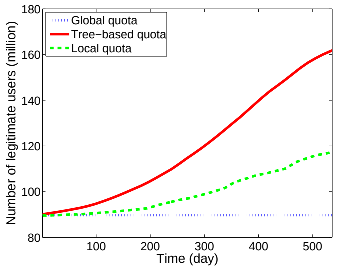
Ideally, we would like all legitimate users to be vouched for eventually. So we first examine the number of GCC-users that can be vouched for over time without attacks. Figure 7 shows the comparisons among the three algorithms.

Without attacks, using the global quota scheme, almost all GCC-users (234 million out of 237 million) can be admitted into the system, including the initial 90 million initial vouchers. In contrast, the local quota system is very restrictive and admits only 119 million users. The tree-based quota scheme performs in between and admits 164 million users, which is 70% of what global quotas can achieve. Since vouching from GCC users can potentially cover 85% all legitimate users (see Section 6.1.1), the tree-based vouching algorithm therefore can admit around $70\% \times 85\% = 60\%$ of all legitimate users assuming there is no negative correlations between the two populations. This is a significant improvement over existing systems. For vouching activities that do not succeed immediately (e.g., because of quotas), we queue the events and fall back to the current practice (e.g., CAPTCHAs) until there exists enough quota.

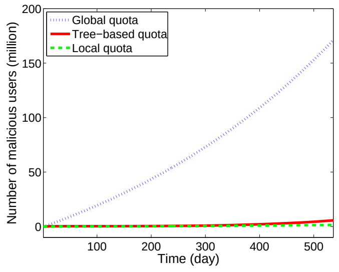
6.2.3 Robustness to Attacks

The real benefit of limiting quota in a community is its robustness to attacks. We inject attacks by randomly picking a subset of users as the compromised users at the beginning of our simulation and they stay malicious ever after. (This scenario is favorable for attackers, as given the fixed growth rate, the longer a user is in the system, the more users it can vouch for over time.) We then generate attack events where compromised users aggressively (i.e., whenever there is quota) vouch for malicious new users, who further aggressively vouch for more malicious users. We repeat each experiment three times with different random seeds and take the average. The results are consistent across runs. Figures 8 (a) and (b) show the population growth of the legitimate and the malicious users over time, assuming 0.5% of the total users are compromised at the beginning.

- For the global quota scheme, in the presence of attacks, the number of legitimate users hardly grows. The reason is that both the compromised users and the new malicious users aggressively use up the system quota, denying the admission of legitimate users. Correspondingly, the malicious user population quickly grows and becomes dominant.
- Using the tree-based scheme, compared with the case without attacks, 99.6% legitimate users are admitted despite the



(a) Legitimate user growth.



(b) Malicious user growth.

Figure 8: Number of legitimate and malicious users admitted into the system over time in the simulated attacks.

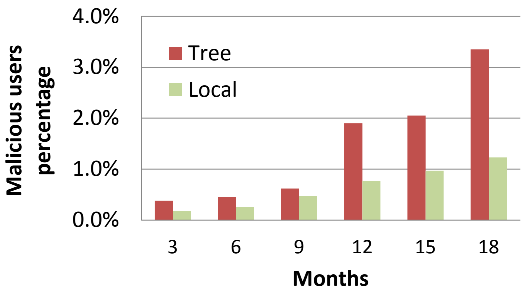


Figure 9: Percentage of malicious users in the system over time. We compare the tree-based quota method with the local quota method.

aggressive behavior from compromised and attacker-created accounts, yet the number of admitted malicious users is comparable to that in the local quota scheme. We examine more closely the tree depths and sizes, and find that over 70% users are on trees of depths between 2 and 7, and around 50% users are on trees ranging from 20 to 40 nodes. The small tree-based community size therefore allows quota sharing among legitimate users while limiting the population growth of malicious users.

- For the local quota scheme, although it admits the smallest number of malicious nodes, the number of legitimate users admitted is also significantly lower than in our tree-based quota mechanism.

Given that the global quota scheme performs significantly worse, we further examine only the tree-based and the local quota schemes in more detail. Figure 9 shows the comparison between the local and the tree-based quota algorithms in terms of the percentage of malicious nodes in the system over time.

With 0.5% of initial compromised users, assuming no malicious nodes were detected, our tree-based quota scheme performs similarly to the local quota scheme during the first 9 months. Such long incubation period may drastically affect the cost model for attackers, which usually targets short-term, immediate returns. If the malicious nodes start being active during the first year, many of them may be detected and traced back, affecting the future growth of the malicious user population.

Finally, Figure 10 shows that as we increase the initial percentage of compromised users (exponentially from 0.25% to 2%), the percentage of malicious users increases roughly proportionally and so the trend is quite predictable and linear. Assuming no detection in place at all, eventually after 18 months, the percentage of malicious users could reach around 12% when 2% of users are initially

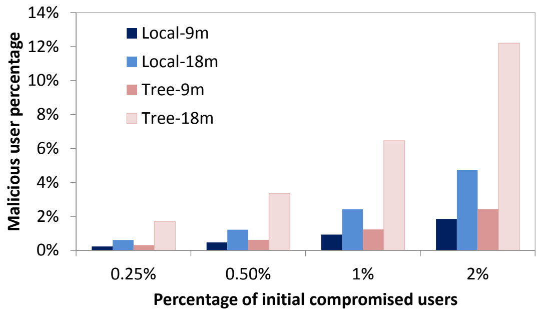


Figure 10: Percentage of malicious users in the system over time by varying the percentage of initial compromised users. We compare the tree-based quota method with the local quota method.

compromised. Note that this is the worst case result assuming no detection in place. In practice, we will not reach this number since many of the malicious accounts will be detected and removed from the system after a few months.

7. DISCUSSION

The essence of our solution is to recognize a large fraction of legitimate new users, while limiting the growth of the malicious user population using quotas. One important challenge is defining scopes of quota sharing and isolation. The global and the local quota schemes are the two extreme ends of the spectrum. We believe the use of quota-sharing trees offers a general intermediate solution, and we demonstrate its effectiveness using a concrete application with large, real data. By adjusting the tree depths and sizes, service providers can flexibly configure the desired degrees of quota sharing to balance security vs. usability based on their specific user growth model and application needs.

The concept of vouching and the use of vouching trees are generally applicable across many applications. Service providers may customize the way of constructing social graphs for bootstrapping based on application semantics. Similarly, they may selectively choose activities that qualify as vouching. Finally, the specific values of our parameters (i.e., the maximum tree size and the vouching delay) may also be tuned according to their datasets.

With Souche in place, attackers may wish to game the system by compromising more accounts, especially those that haven't vouched for many other users and so have more vouching quota. In addition, attackers may also try to compromise users with larger vouching subtrees to share their quota. These strategies are more effective than compromising random accounts or reusing previously compromised accounts. However, they are also more difficult to carry out in practice, as attackers often have no information regarding the vouching tree structures of accounts that are not controlled by them. Furthermore, hijacking specific accounts is always more difficult than hijacking any accounts, particularly those with careless users and hence more susceptible to attacks. Finally, the number of vouched malicious users will still be bounded by the tree size, which can be tightly monitored and limited. For services with high security requirements, Souche can be used in combination with CAPTCHAs and other defenses.

In this paper, we did not focus on detecting attacker-created or compromised vouchers, as Souche is designed to tolerate a small percentage of them and to prevent their population from growing quickly. On the other hand, attack detection is important to ensure that the malicious user population is still bounded in the long term (e.g., one year), and this is a separate topic to address. We believe that the identification and protection of legitimate users should go hand-in-hand with the detection of malicious users in any online service to ensure both security and service usability.

8. CONCLUSION

In this paper, we present Souche, a new system for identifying legitimate users early, soon after they join an online service. Souche is lightweight and can be completely transparent to users. It exploits social connections to gradually expand the legitimate user population through a vouching process based on trust relationships. In our evaluation on a Hotmail dataset, it can recognize the majority of legitimate users early. On the other hand, Souche is effective in bounding the growth of the malicious population. Our results indicate that leveraging social connectivity information offers new opportunities to improve service usability and security.

The presumption of innocence that characterized the early days of online services was beneficial for user experience but, as we noted in the Introduction, it has been largely abandoned in the face of relentless attacks. Our research is an effort to return to a situation where most legitimate users can enjoy online services with few speed-bumps, from the moment that they join those services. In our work, the presumption of innocence is not based on faith but rather on the principle of innocence by association. We demonstrate that this principle has practical validity. At the same time, we can focus security measures on the remaining, suspicious users.

9. REFERENCES

[1] Cyber-Criminals Shift to Compromised Web Mail Accounts for Spam Delivery. `http://www.eweek.com/c/a/Messaging-and-Collaboration/CyberCriminals-Shift-to-Compromised-Web-Mail-Accounts-for-Spam-Delivery-808933/`.

[2] Inside India's CAPTCHA-Solving Economy. `http://blogs.zdnet.com/security/?p=1835`.

[3] Message Bounced Due to Sending Limit. `http://mail.google.com/support/bin/answer.py?hl=en&answer=22839`.

[4] New Spammer Tactics—Compromised Accounts Now Favored. `http://blog.commtouch.com/cafe/data-and-research/new-spammer-tactics`.

[5] Rise in Hacked Gmail, Hotmail, and Yahoo Email. `http://www.boxaid.com/word/viruses-and-malware/rise-in-hacked-gmail-hotmail-and-yahoo-email`.

[6] Spammers Using Porn to Break Captchas. `http://www.schneier.com/blog/archives/2007/11/spammers_using.html`.

[7] Twitter User Reputation Computed from Tweets. `http://blog.tagwalk.com/2009/11/twitter-user-reputation-computed-from-tweets`.

[8] Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. Analysis of Topological Characteristics of Huge Online Social Networking Services. In *WWW*, 2007.

[9] A. Bonato, J. Janssen, and P. Pralat. A Geometric Model for On-line Social Networks. In *Workshop on Online Social Networks (WOSN)*, 2010.

[10] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu. The Socialbot Network: When Bots Socialize for Fame and Money. In *Proc. of the 27th Annual Computer Security Applications Conference (ACSAC'11)*, 2011.

[11] P. Boykin and V. P. Roychowdhury. Leveraging Social Networks to Fight Spam. *IEEE Computer*, 38, 2005.

[12] E. Bursztein, S. Bethard, C. Fabry, J. C. Mitchell, and D. Jurafsky. How Good are Humans at Solving CAPTCHAs? A Large Scale Evaluation. In *IEEE Syposium of Security and Privacy*, 2010.

[13] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro. Aiding the Detection of Fake Accounts in Large Scale Social Online Services. In *NSDI*, 2012.

[14] P. Chirita, J. Diederich, and W. Nejdl. MailRank: Global Attack-Resistant Whitelists for Spam Detection. In *Conference on Information and Knowledge Management (CIKM)*, 2005.

[15] G. Danezis and P. Mittal. SybilInfer: Detecting Sybil Nodes using Social Networks. In *NDSS*, 2009.

[16] J. Douceur. The Sybil Attack. In *IPTPS*, 2002.

[17] J. Golbeck. *Computing with Social Trust*. Springer, 2008.

[18] C. Grier, , K. Thomas, V. Paxson, and M. Zhang. @spam: The Underground on 140 Characters or Less. In *CCS*, 2010.

[19] S. Hao, N. A. Syed, N. Feamster, A. G. Gray, and S. Krasser. Detecting Spammers with SNARE: Spatio-temporal Network-level Automatic Reputation Engine. In *USENIX Security*, 2009.

[20] J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *Proc. 32nd ACM Symposium on Theory of Computing*, 2000.

[21] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and Analysis of Online Social Networks. In *IMC*, 2007.

[22] A. Mohaisen, A. Yun, and Y. Kim. Measuring the Mixing Time of Social Graphs. In *IMC*, 2010.

[23] 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 *Usenix Security*, 2010.

[24] A. P. V. Shah and A. Mislove. Bazaar: Strengthening User Reputations in Online Marketplaces. In *NSDI*, 2011.

[25] N. Tran, J. Li, L. Subramanian, and S. S. Chow. Optimal Sybil-resilient Node Admission Control. In *Infocom*, 2011.

[26] N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-Resilient Online Content Voting. In *NSDI*, 2009.

[27] B. Viswanath, K. P. Gummadi, A. Post, and A. Mislove. An Analysis of Social Network-Based Sybil Defenses. In *SIGCOMM*, 2010.

[28] C. Wilson, B. Boe, A. Sala, K. P. N. Puttaswamy, and B. Y. Zhao. User Interactions in Social Networks and their Implications. In *EuroSys*, 2009.

[29] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov. Spamming Botnets: Signatures and Characteristics. In *SIGCOMM*, 2008.

[30] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai. Uncovering Social Network Sybils in the Wild. In *IMC*, 2011.

[31] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks. In *IEEE Symposium on Security and Privacy*, 2008.

[32] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: Defending Against Sybil Attacks via Social Networks. In *SIGCOMM*, 2006.

[33] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. K. Gunda, and J. Currey. DryadLINQ: A System for

General-Purpose Distributed Data-Parallel Computing Using a High-Level Language. In *OSDI*, 2008.

[34] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, and E. Gillum. BotGraph: Large Scale Spamming Botnet Detection. In *NSDI*, 2009.

APPENDIX

A. TWO PROPERTIES OF THE VOUCHING ALGORITHM

Here we sketch proofs for the two properties presented in Section 5.4.2:

User population growth property: If the growth rate is a constant r, then the global user growth rate across all vouching trees will be no more than r.

This property is easy to prove. For each newly vouched node, there is always at least one unit of quota from the vouching tree available to support it. And with quota debit, the sum of the node quota on a vouching tree equals the sum of the node quota computed individually using the local quota algorithm with rate r. Thus the entire system growth rate will be no more than r.

Malicious-user population growth property: In a simplified scenario where trees are of the same height and every node behaves identically, i.e., has the same out-degree and vouches for new nodes with rate r exactly, then with α fraction of nodes being compromised and uniformly distributed in the system, the fraction of admitted nodes that are malicious will be no more than $(k-1)\alpha$ at any time, where k is the depth of the trees at the time when the malicious nodes first start vouching.

To analyze the fraction of admitted nodes that are malicious, we first consider where the compromised nodes might locate on the vouching trees. With uniform distribution, the fraction of compromised nodes at each level of the trees will all be α. When a node is compromised at tree level l, we consider the worst case, where all the quota on the corresponding vouching subtree rooted at its parent at level $l-1$ will be used up by malicious nodes. In the case of a root node being compromised, the vouching subtree is the entire tree.

Let Q_l represent the quota of all vouching subtrees starting at level l (with $l=1$ being the root, and $Q_1 = Q$). Let Q' represent the percentage of total global quota that can be exploited by malicious nodes, and Q'_l represent the percentage of global quota that can be exploited by malicious nodes locating at level l. Among all the nodes at level i, $(1-\alpha)^{(i-1)}$ of them locate outside the subtrees of compromised nodes at levels smaller than i. So the fraction of additionally compromised nodes at level i is $\alpha(1-\alpha)^{(i-1)}$. Therefore,

$$\begin{aligned} Q' &= Q'_1 + Q'_2 + Q'_3 + \cdots + Q'_{k-1} \\ &= \alpha Q_1 + \alpha(1-\alpha)Q_2 + \ldots + \alpha(1-\alpha)^{k-2}Q_{k-1} \\ &\le Q\alpha \sum_{i=0}^{k-2}(1-\alpha)^i \\ &\le (k-1)\alpha Q \end{aligned}$$

B. EFFECTIVENESS OF THE OPTIMIZATION HEURISTICS

We study the effectiveness of our optimization heuristics as described in Section 5.4.3. We examine the user population increase with and without optimizations and plot their ratio in Figure 11. We find that the optimization strategies have much more impact on the malicious node population than on the legitimate nodes. Without our heuristics, the number of malicious nodes is 1.85 times that with heuristics, and the relative malicious population percentage increased by 55%. Hence the optimization heuristics are effective in throttling the population growth of malicious users. They penalize aggressive behavior without significantly affecting legitimate users.

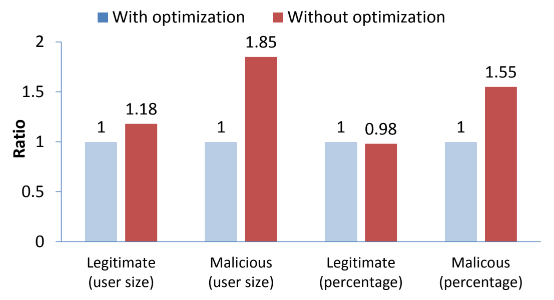


Figure 11: The ratio of user (population) size increase without optimization heuristics compared to that with optimization. We examine both the user size increase and the percentage increase.