10.1109/TDSC.2015.2410792, IEEE Transactions on Dependable and Secure Computing

VoteTrust: Leveraging Friend Invitation Graph to Defend against Social Network Sybils

Zhi Yang, Jilong Xue, Xiaoyong Yang, Xiao Wang, and Yafei Dai

Abstract—Online social networks (OSNs) suffer from the creation of fake accounts that introduce fake product reviews, malware and spam. Existing defenses focus on using the social graph structure to isolate fakes. However, our work shows that Sybils could befriend a large number of real users, invalidating the assumption behind social-graph-based detection. In this paper, we present VoteTrust, a scalable defense system that further leverages user-level activities. VoteTrust models the friend invitation interactions among users as a directed, signed graph, and uses two key mechanisms to detect Sybils over the graph: a voting-based Sybil detection to find Sybils that users vote to reject, and a Sybil community detection to find other colluding Sybils around identified Sybils. Through evaluating on Renren social network, we show that VoteTrust is able to prevent Sybils from generating many unsolicited friend requests. We also deploy VoteTrust in Renen, and our real experience demonstrates that VoteTrust can detect large-scale collusion among Sybils.

Index Terms—Online Social Network, Sybil Attack, Sybil Detection, Spam

1 Introduction

Recently, OSNs have come under Sybil attacks [1]. In this attack, a malicious user creates multiple fake identities, known as *Sybils* [1], to unfairly increase their power and influence within a target community. Researchers have observed Sybils forwarding spam and malware on Renren [2], Facebook [3] and Twitter [4].

To defend against Sybils, prior Sybil defenses [5]–[9] leverage the positive *trust* relationships among users, and rely on the key assumption that Sybils can befriend only few real accounts [10]. Unfortunately, we find that people in real OSNs still have a non-zero probability to accept friend requests of strangers, leaving room for Sybils to connect real users through sending a large amount of requests.

In this paper, we further explores the negative *distrust* relationships (e.g., in the form of rejected friend requests) among users, as Sybils have more distrust relationships than trust ones with real users. However, this feature cannot be directly applied because attackers could obfuscate their Sybils from the detector by generating many fake trust relationships among Sybils.

To prune the fake relationships, we model the friend invitation interactions among users as a signed, directed network, with an edge directed from the sender to the receiver and a sign $(1/-1)$ indicates whether a friend request is accepted. This graph is referred to as the *friend invitation graph*, as illustrated in Fig. 1. The fundamental rationale of our approach is to leverage the unique structural features of Sybil community in this signed graph: the colluding Sybils as a whole has a limited number of incoming links and more negative outgoing links than positive ones, since real users usually send/accept the friend requests to/from their friends or acquaintances.

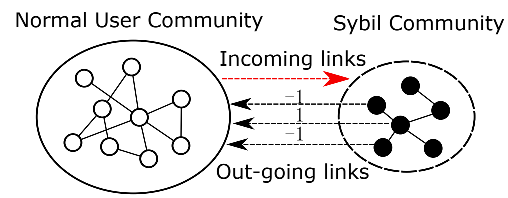


Fig. 1. Illustration of the friend invitation graph and the structure of Sybil community .

Based on the above rationale, we present *VoteTrust*, a system that leverages the friend invitation graph to detect Sybils. In VoteTrust, we say that a node B casts a (positive/negative) vote on a node A if B accepts/rejects the request from A. VoteTrust first uses a PageRank-style algorithm to appropriately assign the number of votes that one can cast on another node (referred to as *vote capacity*). This process assigns few vote capacity for individual Sybils and thus prevents them from significantly vouching each other through collusion. After that, VoteTrust evaluates a *global* acceptance rate (i.e., the probability of being a real user) for each node through aggregating the votes over the network. During the aggregation, VoteTrust further penalizes votes from suspected nodes. Due to more negative votes from real users, Sybils would get low global acceptance rates and thus can be identified out.

This paper significantly extends an earlier version [11] in the following ways. First, we add a new Section 2 to characterize the friend request behavior of Sybils. Second, we add a new Section 5 that looks at how to detect Sybil communities surrounding the identified Sybils. Third, we add a new Section 6.2.2 that looks at

- *Zhi Yang, Jilong Xue, and Yafei Dai are with the Department of Electrical and Computer Engineering, Peking University, Beijing 80071, P.R. China. E-mail: {yangzhi, xjl, dyf}@net.pku.edu.cn*
- *Xiao Wang and Xiaoyong Yang is with Renren Inc, Beijing, China. E-mail:{xiao.wang, xiaoyong.yang}@renren-inc.com*

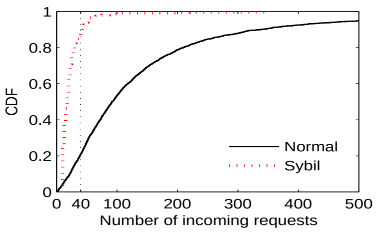


Fig. 3. The number of incoming friend requests

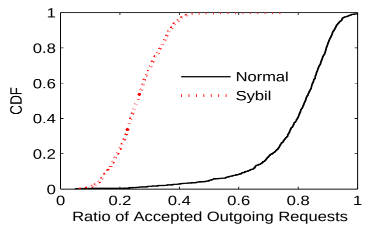


Fig. 4. Ratio of accepted outgoing friend requests

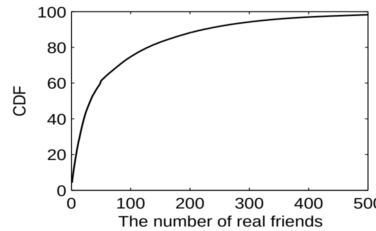


Fig. 5. The number of real users the Sybils have befriended

Fig. 2. A sample of Sybil accounts.

the performance gain of the Sybil community detection component. Finally, we add a new Section 7 to describe how to implement VoteTrust using parallel computing frameworks (e.g., Giraph), and report on our real experiences with a deployment of VoteTrust in Renren.

2 Background

In order to reach a user on OSNs like Renren and Facebook, the attacker must first befriend that user. This is because, by default, social communications such as creating posts are only allowed between friends. The Sybils cannot be monetized without first establishing social connections to real users. This motivates us to exploit the friend request behavior to detect Sybils.

To help, Renren [2], [12], one of the most popular OSNs in China, provided us with 1000 real accounts and 1000 Sybil accounts, respectively. Both Sybils and real users are sampled from over the global Renren network, which ensures the identified features are typical (e.g., not special features of users in a regional network). The Sybils were confirmed spammers due to containing spam in blogs or posts. We find that most Sybil accounts are well-constructed. Fig. 2 is the profile of a sample account. The profile is crafted as a college student with pictures, and with blogs and friends.

To avoid the effect of fake relationships, Renren sent CAPTCHAs to all the friends of Sybils, and only considered those passing the challenge as real friends. The security team of Renren also helped us to inspect a random sample of 100 nodes from those passing the test and confirmed that 96% of them are real users.

2.1 Befriending Behavior of Sybils

Using this dataset as our ground truth, we made the following key observations:

First, Sybils receive few incoming requests from real users. Fig. 3 plots a cumulative distribution function (CDF) of users by the number of incoming requests they receive. We see that Sybils receive few friend requests from real users, since real users are more like to send requests to their real life acquaintances or friends.

Second, Sybils are more likely to receive rejections than real users. Fig. 4 shows a distinct difference between Sybils and real users in terms of the acceptance rate, i.e., the fraction of outgoing friend requests accepted by real users. On average, real users have a high acceptance rate of 0.8, whereas Sybils have a low acceptance rate of 0.2. Interestingly, the average acceptance rate of Renren Sybils is close to that of fictitious profiles in Facebook [13], indicating that the users of different OSNs have a similar degree of tolerance to unwanted communication.

Since Sybils have non-zero acceptance rates, they can befriend many real users by sending a large number of friend requests. Fig. 5 shows the CDF of the number of Sybils' friends passing the CAPTCHA test. We find that 50% of Sybils have more than 32 real friends. On average, each Sybil has about 65 friends passing the test.

2.2 Discussion

What is the key difficulty of Sybils? The current social-graph-based Sybil defenses assume that the key difficulty of Sybils is to befriend many real users [5]–[9]. However, our results show that Sybils can easily overcome this difficulty by sending a large amount friend requests. Their actual difficulty is to require real users to befriend them first or to accept them with a high probability.

Can we directly use this difficulty to detect Sybils? Suppose that we detect Sybils with the structural features of the incoming degree or the percentage of positive outgoing links. An attacker could manipulate these features with Sybil collusion: One Sybil can send friend requests to other colluding Sybils, who are guaranteed to accept these requests. This is the reason why we propose VoteTrust algorithm which is more robust against manipulations of the graph.

3 Models and Goal

In this section, we outline the system and threat models and the goal of VoteTrust.

System model. We consider a social network (like Renren and Facebook) that adopts a friend request/confirm mechanism. One has to send a request in order to befriend another user, and the recipient can accept or reject the request. We model the request/confirm interactions of users as a **friend invitation graph**: a directed and signed graph $G(V, E)$, where V and E are the set of nodes and links, respectively. A link $e = (u, v, s)$ from u to v, of sign $s = 1$, indicates that v trusts u and accepts its request. If $s = -1$, then v distrusts u and rejects its request. Let E^+ and E^- are disjoint sets of positive and negative links ($E^+ \cup E^- = E$).

In the graph, the node set V contains two disjoint sets H and S, representing real and Sybil users respectively. We denote the real region G_H as the subgraph that includes all real users and the links among them, and the Sybil region G_S as the subgraph that includes all Sybils and the links among them. Since real users are not likely to send/accept the friend request to/from strangers such as fake accounts, G_S has few incoming links from G_H, but more negative outgoing links than positive ones to G_H. In this paper, we use the term **In-link** to represent the link that goes *into* the Sybil region G_S from the real region G_H.

Attack model. To appear legitimate to the system, an attacker could create many positive links among Sybils. The objective of the attacker is to infiltrate the target OSN by creating as many links as possible to the real region. We use the term **attack-link** to represent the link that goes from the Sybil region G_S to the real region G_H.

Goal. The goal of VoteTrust is to takes as input the friend invitation graph G, and outputs the classification of any node u, i.e., $u \rightarrow \{$*real*, *Sybil* or *unknown*$\}, \forall u \in V$. When a node u joins the network, its initial state is *unknown*. However, as the node repeatedly sends requests to normal users, the system can eventually classify it as *Sybil* or *real* based on the feedbacks from real users. Let N_{out} be the maximum number of attack-links that a Sybil can create before being detected. VoteTrust aims to ensure that N_{out} is bounded, irrespective of the number of collusion links.

4 Individual Sybil Detection

We now describe the design of VoteTrust, which considers the Sybil detection as a vote aggregation problem. In VoteTrust, a link of the friend invitation graph means that one node casts a certain number of votes for the other. The vote value is determined by the sign of link. For each node, VoteTrust guarantees that votes are mainly collected from real users by pruning the collusion votes among Sybils. Then, it can identify the Sybil for which the majority of votes are negative.

In VoteTrust, each node has two important features: i)**Vote capacity** $\vartheta(v)$ is the number of votes that v can cast on another node. Given a link (u, v, s) in G, we consider that node v casts at most $\vartheta(v)$ votes on node u, and the vote value $x_{vu} = s$ (-1 or 1). ii) **Global acceptance rate** $p(u)$ is the fraction of positive votes that VoteTrust aggregates for a node u, indicating the probability that u is accepted by real users. Nodes with low global acceptance rate (e.g., below a certain threshold δ_f) are detected as Sybils. To limit the Sybil collusion, VoteTrust uses two key techniques, *trust-based vote assignment* and *global vote aggregating*, to properly assign the vote capacity and to compute the global acceptance rate.

4.1 Trust-based Votes Assignment

The goal of trust-based votes assignment is to assign low vote capacity to Sybils, so that we can limit the number of votes that Sybils could cast for each other. To achieve this goal, we first select some trusted users as seeds, and then propagate the vote capacity from the seeds to others along the links of friend invitation graph $G(V, E)$. As Sybil region has a limited number of in-links, the total vote capacity entering the Sybil region is constrained.

Selecting Trusted Seeds. The goal of seed selection is to find real users that will be the most useful in identifying other real users. A heuristic for selecting seeds is to give preference to those from which trust can be propagated to many other real users. Note that real users prefer to send requests to their real-life acquaintances, so we use the inverse PageRank method like TrustRank [15]. The basic idea is to build the seed set from real users that point to many real users that in turn point to many others and so on. In particular, we can reverse the links in the friend invitation graph, and compute the PageRank. Through manually inspecting a few users of high inverse PageRank scores, OSN providers can easily identify those real users to seed trust.

Suppose that the system has $N = |V|$ vote capacity in total, i.e., each node has one vote capacity on average. Given a set of trusted seeds (denoted as V_s), we equally assign the vote capacity over V_s. Thus the initial vote capacity for a user u is,

$$\mathbf{I}(u) = \begin{cases} N/|V_s|, & \text{if } u \in V_s; \\ 0, & \text{otherwise.} \end{cases}$$

Votes Propagation. We then propagate the vote capacity from trusted seeds to other nodes as follows: Suppose that each incoming neighbor v of a node u has a vote capacity of $\boldsymbol{\vartheta}(v)$ and a outgoing degree of $\omega(v)$. The node u's overall vote capacity can be computed as,

$$\boldsymbol{\vartheta}(u) = d \cdot \sum_{v:(v,u) \in E} \frac{\boldsymbol{\vartheta}(v)}{\omega(v)} + (1 - d) \cdot \mathbf{I}(u), \tag{1}$$

where d is a constant less than 1 (e.g., 0.8).

4.2 Global Vote Aggregating

Vote assignment gives low vote capacity to not only Sybils but also non-popular real users with few incoming links. We thus introduce the global vote aggregating

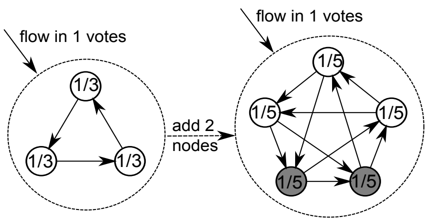


Fig. 6. Example of vote collection

phase to get the global acceptance rate $p(u)$ of a node u. This phase further leverages the sign of outgoing links (i.e., the user feedback) for higher accuracy, as Sybils have a higher percentage of negative links to real region.

Global rating computation. For a node u, VoteTrust computes the $p(u)$ by combining all the votes from its outgoing neighbors. As neighbors of high global acceptance rates are more likely to be real users, we should bias towards their votes. Based on the above intuition, we compute the user u's global acceptance rate $\boldsymbol{p(u)}$ as:

$$\widehat{\boldsymbol{p}}(u) = \frac{\sum_{v:(u,v)\in E^+} \boldsymbol{\vartheta}(v) \cdot \boldsymbol{p}(v)}{\sum_{v:(u,v)\in E} \boldsymbol{\vartheta}(v) \cdot \boldsymbol{p}(v)}, \tag{2}$$

Notice that the voting result may be unreliable if there are few votes casted for a node. Hence, we combine Wilson score to increase the confidence. Suppose the number of votes for u is $n(u)$. The Wilson score is a weighted average of $\widehat{\boldsymbol{p}}$ and 0.5, with $\widehat{\boldsymbol{p}}$ receiving greater weight as $n(u)$ increases, i.e.,

$$\boldsymbol{p}(u) = \frac{\widehat{\boldsymbol{p}}(u) + \frac{1}{2n(u)} z^2_{1-\alpha/2}}{1 + \frac{1}{n(u)} z^2_{1-\alpha/2}}, \tag{3}$$

where the $z_{1-\alpha/2}$ is the $1-\alpha/2$ percentile of a standard normal distribution, e.g., $z_{1-\alpha/2} = 1.96$ for 95% confidence level.

Limiting the collusion votes. When aggregating votes of outgoing neighbors, an important problem we should address is *how to prevent the attacker from increasing the total number of collusion votes by enlarging the Sybil set?*

Considering the case illustrated in Fig. 6. Initially, the Sybil region has 3 Sybils that receive a total of 1 vote capacity from the real region. The vote capacity of each Sybil is $1/3$, and each Sybil can collect at most $1/3$ collusion votes. However, if the attacker adds another two Sybils, the vote capacity of individuals drops to $1/5$ as the total vote capacity is constant. But each Sybil can collect at most $2/5$ collusion votes. This means that the attacker can increase collusion votes for Sybils by enlarging the Sybil region. In fact, a complete-connected subgraph with N Sybils and c total capacity could create $\frac{c(N-1)}{2}$ collusion votes, which increases as N grows.

Given a fixed number of in-links, the vote capacity of individual Sybils will drop as the Sybil region is enlarged. Thus, VoteTrust limits the size of Sybil region by ignoring the votes from nodes of very low capacity, i.e, below a certain threshold δ_v. For example, in Fig. 6, if we set $\delta_v = 1/3$, all the collusion votes would be ignored once the individual capacity drops to $1/5$ due to adding more Sybils. Choosing the threshold δ_v should make a balance between ignoring collusion votes within Sybil community and losing some real votes. We shall show how to make a balanced tradeoff in Section 6.

---

procedure VOTETRUST-D(G, V_s)
    if $u \in V_s$ **then** $\triangleright$ vote assignment
        $\mathbf{I}(u) \leftarrow N/|V_s|$;
    else
        $\mathbf{I}(u) \leftarrow 0$;
    end if
    while $\Delta > \varepsilon_1$ **do**
        for $u \in V$ **do**
            $\boldsymbol{\vartheta}(u) = d \cdot \sum_{v:(v,u)\in E} \frac{\boldsymbol{\vartheta}(v)}{\omega(v)} + (1-d) \cdot \mathbf{I}(u)$
        end for
    end while
    $\boldsymbol{p}^{(0)} \leftarrow 0.5$; $\triangleright$ vote aggregating
    while $\Delta > \varepsilon_2$ **do**
        for $u \in V$ **do**
            $\widehat{\boldsymbol{p}}(u) = \frac{\sum_{v:(v,u)\in E^+} \boldsymbol{\vartheta}(v)\cdot\boldsymbol{p}(v)}{\sum_{v:(v,u)\in E} \boldsymbol{\vartheta}(v)\cdot\boldsymbol{p}(v)}$
            $\boldsymbol{p} \leftarrow$WilsonScore($\widehat{\boldsymbol{p}}$);
        end for
    end while
end procedure

Fig. 7. Sybil detection algorithm of VoteTrust.

Sybil detection. Given a detection threshold δ_f, we consider a node u as Sybil if its global acceptance rate $p(u) < \delta_f$. Before combining votes using equation (2), we assign the initial value of $\boldsymbol{p}(u)$ as δ_f. For those without any vote (e.g., new users), their global rating would always be the initial value. Thus, the system does not perform classification on newcomers until after they start generating friend requests.

We sketch the VoteTrust system in Fig. 7. It takes the friend invitation graph G and a set of trusted "seed users" V_s as inputs, and outputs a set of active Sybils that send many friend requests to real users.

4.3 Security properties

let N_{in} be the number of in-links entering the Sybil region G_S, and N_{out} be the number of attack-links a Sybil can send to real users. VoteTrust provides the following security guarantees:

Theorem 1: If a colluding Sybil s wants to keep its global acceptance rate $p(s)$ above the detection threshold δ_f, the number of its attack-links N_{out} should follow:

$$N_{out} \leq \rho \frac{\delta_f - \delta_f^2}{\delta_f - \gamma} N_{in}, \tag{4}$$

where ρ is a constant and γ is the fraction of negative attack links of Sybils.

The proof is in Appendix A. This theorem means that the number of requests that a Sybil could send to real users are linearly bounded by the number of requests it receives from real users.

Theorem 2: The system collects collusion votes of Sybil region S only if the Sybil group size $N_s = |S|$ satisfies:

$$N_s \leq \sigma \frac{N_{in}}{\delta_v}, \tag{5}$$

where σ is a constant factor and δ_v is the threshold for vote collection.

The proof is in Appendix B. We see that the size of Sybil community is also constrained by the number of in-links N_{in}.

5 SYBIL COMMUNITY DETECTION

Detecting Sybil community in friend invitation graph is different from that in social graph. In social graph, given a node u and a community C, an internal (or external) link indicates that the node belongs (or does not belong) to the community. So the node u is included in C only when it has more internal links than external ones. Hence, Sybils do not form tight-knit communities if they are not well connected among each other, or accumulate many external edges from real users, as demonstrated by our prior work [2]. As a result, they may be classified as real users belonging to nearby real communities.

Unlike social graph, friend invitation graph is a **signed** network that further contains the negative links. Given a node u and a community C, both internal positive links and external negative link indicate that the node belongs to the community. Although Sybils are not well connected (i.e., few internal positive links) or accumulate many external positive links, they also have many external negative links to indicate their community membership. As a result, Sybils form a community in friend invitation graph, meaning group members (e.g., Sybils) accept each other but are rejected by the nodes out of the group (e.g., real users).

Given a known Sybil, the next goal of VoteTrust is to detect the whole Sybil community in the friend invitation graph. Once some Sybil nodes are identified using the voting-based detection, VoteTrust begins to detect the Sybil community around these Sybil seeds. VoteTrust employs two key techniques, *bad score propagation* and *Sybil community identification*, to correctly expand Sybil community and to determine its boundary.

5.1 Bad Score Propagation.

The goal of bad score propagation is to find nodes that are more likely to be colluders. It assigns each node a bad score, and a large score indicates a high likelihood of being Sybil. This score assignment can guide us to identify the boundary of Sybil region in the next step.

In order to assign high bad score to Sybils, VoteTrust propagates the bad score from some identified "seed Sybils" to other nodes along the *inverse* direction of links. Since the Sybil region has a fixed number of in-links, the bad score is likely to stay within the subgraph consisting of colluding Sybils.

Selecting Bad Seeds. Define $l(s)$ as the local acceptance rate of a Sybil s, i.e., the percentage of its positive links. Clearly, if a Sybil s has many colluding Sybil neighbors, its local acceptance rate $l(s)$ would be larger than its global acceptance rate $p(s)$. Hence, given a set of Sybil nodes, we use the subset $V'_s = \{s|p(s) < l(s)\}$ as seeds, and equally assign the initial bad score over them.

$$\mathbf{I}(u) = \begin{cases} 1, & \text{if } u \in V'_s. \\ 0, & \text{otherwise.} \end{cases}$$

Bad Score Propagation. Next we propagate bad score from Sybil seeds to others along the inverse direction of links in G: Suppose each outgoing neighbor v of a node u has a bad score of $\boldsymbol{B}(v)$ and a in-degree of $\varphi(v)$. The node u's bad score can be computed as,

$$\boldsymbol{B}(u) = d \cdot \sum_{v:(u,v)\in E} \frac{\boldsymbol{B}(v)}{\varphi(v)} + (1-d) \cdot \mathbf{I}(u). \tag{6}$$

5.2 Sybil Community Identification

Sybil community identification aims to identify the Sybil community boundary. Instead of detecting the community on the entire graph, we detect a local Sybil community C by expanding a known portion of the community. This portion C_0 can be thought of as the *core* of a local community. The algorithm first includes a number of nodes with the highest bad score as the community core C_0, and then detects the Sybil community C through a node discovery process: We focus on the two local regions of the network. One is the current community C (initial state is C_0), and the other is the set of nodes adjacent to the community, D (each has at least one incoming or outgoing neighbor in C). At each step, one or more nodes from D are chosen and agglomerated into C, then D is updated to include any newly discovered nodes. This expansion process continues until it has discovered the entire local community.

Local Community Expansion To determine which nodes should be agglomerated into the community at each step, we measure to what extent a node $u \in D$ is accepted by the Sybil community C by:

$$\Omega_{in}(u) = \sum_{v\in C} sign(u,v) \tag{7}$$

Similarity, we measure to what extent u is accepted by nodes out of the community C by:

$$\Omega_{out}(u) = \sum_{v\notin C} sign(u,v) \tag{8}$$

Since finding a community corresponds to increasing its internal positive links and external negative ones, *we agglomerate the node u with $\Omega(u) = \Omega_{in}(u) - \Omega_{out}(u) > 0$,* i.e., nodes are more likely to be accepted by the nodes in the community than by those out of the community.

Core Selection and Expansion Termination In order to expand the community, the algorithm needs to carefully select the community core C_0. Intuitively, a small

1: **procedure** VOTETRUST-C(G, V_s')
2:   **if** $u \in V_s'$ **then** ▷ Bad score propagation
3:     $\mathbf{I}(u) \leftarrow 1$;
4:   **else**
5:     $\mathbf{I}(u) \leftarrow 0$;
6:   **end if**
7:   **while** $\Delta > \varepsilon_1$ **do**
8:     **for** $u \in V$ **do**
9:       $\boldsymbol{B}(u) = d \cdot \sum_{v:(u,v)\in E} \frac{\boldsymbol{B}(v)}{\varphi(v)} + (1-d)\cdot \mathbf{I}(u)$
10:     **end for**
11:   **end while**
12:   $C_0 \leftarrow V_s'$; ▷ Community expansion
13:   **repeat**
14:     $C \leftarrow$ TopBadScoreNodes($2|C_0|$);
15:     **for** $u \in C$ **do** ▷ Pruning real users
16:       **if** $\Omega(u) < 0$ **then**
17:         $C = C - \{u\}$;
18:       **end if**
19:     **end for**
20:     **for** $u \in D$ **do** ▷ Adding other Sybils
21:       **if** $\Omega(u) > 0$ **then**
22:         $C = C \cup \{u\}$;
23:       **end if**
24:     **end for**
25:     $\Delta \leftarrow 1 - \frac{|C_0|}{|C|}$;
26:     $C_0 \leftarrow C$;
27:   **until** $|C_0| > 0 \& \Delta < \varepsilon_2$
28: **end procedure**

Fig. 8. Algorithm of E-VoteTrust.

core cannot recall many Sybils, whereas a large core may exceed the boundary of Sybil region.

Our algorithm attempts to choose C_0 in a self-adjusting manner. Let m be the size of C_0 in the current iteration (initially, $C_0 = V_s'$). The algorithm first expands the community C from C_0. Then, the algorithm starts a new iteration, doubles the size of C_0 by including top $2m$ nodes with the highest bad score, and re-expands the community C. This process is repeated until the expanded Sybil communities are similar at the end of two consecutive iterations, implying the whole Sybil community is enclosed. Then the algorithm outputs C as the detected sybil community.

The basic ideas of core selection are as follows: First, we select core nodes from those of highest bad scores, expanding the community from the Sybil region. Second, we double the size of C_0 in each iteration, which allows the searching to be efficient and avoids introducing many real users in one iteration.

We sketch the community detection algorithm in Fig. 8. It takes the invitation graph G and a known Sybil set V_s' as inputs, and outputs the Sybil community C that Sybil nodes in V_s' belong to. Notice that some real users could also have relatively high bad score due to unintentionally sending friend requests to Sybils. These nodes may be included in C_0. Thus, we first remove nodes that are more likely to be accepted by the nodes out of C (e.g., from step 15 to 19). This prunes some real users and enhances the accuracy in the subsequent Sybil community expansion.

5.3 Security Property

Theorem 3: The number of real users whose bad score are higher than the Sybils' average score is linearly bounded by the number of In-links N_{in}:

$$|\{u|x_u > \bar{x}_s\}| < \varrho N_{in} \tag{9}$$

where x_u, x_s represent the average bad score of real users and Sybils, respectively, and ϱ is a constant.

The proof is in Appendix C. Recall that the VoteTrust tries to find a *Sybil community* surrounding Sybil seeds, in which each node is detected as a Sybil colluder. A false positive occurs only if we include a real user in the Sybil community. Since the VoteTrust selects nodes of high bad scores (i.e., those likely-to-be-Sybils), the above property guarantees that most nodes in the Sybil community are true Sybils. Thus, the VoteTrust could rightly expand the community *from the Sybil region*, and could iteratively improve the precision (and recall) by removing real users from the community (and by adding other Sybils into the community).

6 EVALUATION

In this section, we evaluate the performance of VoteTrust through conducting detection over Renren network.

6.1 Data Set and Methodology

As VoteTrust requires the structure of friend invitation graph, the sampled dataset used in Section 2 cannot be used for the evaluation. In our experiment, we use the Peking university (PKU) network because: First, we can construct the friend invitation graph for PKU network as we have the complete friend request records (a total of 5.01 million for about 230K PKU users). Second, PKU network is one of the most popular regional networks in Renren, making it an attractive target for attackers (e.g., Sybils are disguised as PKU users to increase their popularity). Finally, we can perform a comparative evaluation of VoteTrust's ability over the PKU network, as we can perform manual inspection on PKU users [1].

We evaluate the performance of VoteTrust through i) adding artificial Sybils and ii) detecting real Sybils in the network. The two sets of Sybils serve to validate different aspects of VoteTrust. The simulations based on artificial Sybils validate whether our theoretical bounds hold in different attacking cases (e.g., different number of in-links or Sybils). The detection performed on real Sybils compares VoteTrust against other approaches in terms of its effectiveness in identifying real social Sybils.

To measure the detection performance, we use the metrics *Precision* and *Recall*. Suppose that O represents the set of *true* Sybils existing in the network, and I represents the set of Sybils *identified* by a defense scheme. Precision $P = |I \cap O|/|I|$ and recall $R = |I \cap O|/|O|$.

1. Renren's privacy policy allows users in the same affiliation to see the profile information of each other.

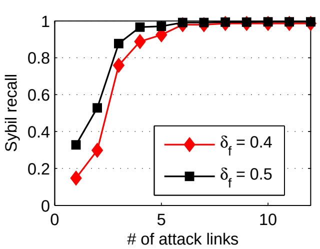


Fig. 9. Sybil recall vs number of out-links

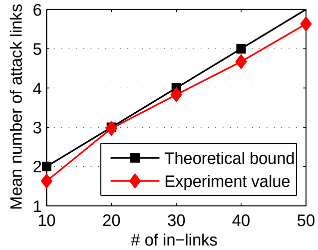


Fig. 10. The upper bound of out-links for Sybils

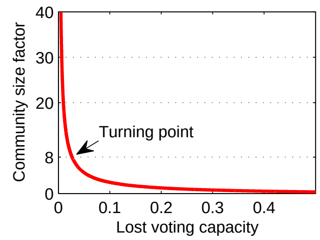


Fig. 11. Tradeoff between vote capacity loss and community size factor

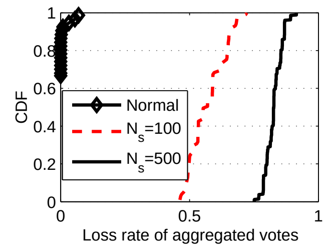


Fig. 12. Loss rate of aggregated votes for normal users and Sybil community

6.2 Detecting Simulated Sybils

We first examine the performance of VoteTrust under various attack strategies. We demonstrate that VoteTrust is able to limit the attack-links of Sybils, and to find other colluding Sybils in a preventive manner.

6.2.1 Sybil Detection

Simulation setup. We create a Sybil region $S = (N_s, N_{in}, N_{out}, p)$, where N_s is number of Sybils, N_{in} is the number of in-links of Sybil region, N_{out} is the number of attack links of each Sybil and p is the percentage of negative attack links. Clearly, the parameters N_{in}, p represent the resource limitation of the attacker, and the goal of VoteTrust is to limit both N_{out} and N_s. In the following experiments, we assume that Sybils form a tight complete-graph to maximize the collusion votes (the worst case). We set $p = 0.2$ according to the measurement in Section 2.

Limiting attack-links N_{out}. We first fix $N_s = 100$ and $N_{in} = 10$, and allow varying attack-links to randomly selected PKU users. Fig. 9 shows the recall increases as N_{out} grows. On average, individual Sybils could only create $E[N_{out}] = 2$ attack-links before being detected, which is much smaller than N_{in}. Notice that the number of PKU users with acceptance rates below a given threshold is constant, so the precision also increases as N_{out} grows due to finding more simulated Sybils. However, we do not evaluate the precision here because there are many real Sybils in Renren, which would affect the true precision of detecting simulated Sybils. We shall leave the detection of real Sybils in Section 6.3.

To verify the bound (4) on attack-links, we vary the number of in-links N_{in}, and compute the average number of attack-links $E[N_{out}]$ that a Sybil can create before being detected. Fig. 10 plots the theoretical upper bound and the experiment result, which are very close.

Limiting Sybil region size N_s. To limit the size of Sybil community, VoteTrust ignores the votes from users with vote capacities below the threshold δ_v. From bound (5) we know that increasing the threshold could reduce the community size N_s to $\rho N_{in}/\delta_v$, but at the *cost* of losing votes from some real users. Fig. 11 shows the trade-off curve for various thresholds, where Y-axis is the reduction factor $1/\delta_v$ and X-axis is the ratio of lost votes to total votes (computed based on vote capacity distribution). To make a balance between vote loss and community size restriction, we select the turning point of $(0.01, 8)$, and get the corresponding threshold $\delta_v = 0.12$.

Given this threshold, we measure the fraction of lost real votes and the fraction of collusion votes ignored under different Sybil community sizes N_s. Here, we fix $N_{in} = 50$ and $N_{out} = 10$. Fig. 12 shows that the normal users lose only 0.45% votes, whereas Sybils lose majority of their collusion votes, e.g., losing 57.1% given $N_s = 100$ and 82.8% given $N_s = 500$. This result indicates that, as the community size grows, VoteTrust effectively eliminates colluding votes within Sybil community while incurring little effect on real users.

6.2.2 Community Detection

We now examine to what extent that the community detection (CD) of VoteTrust could complement the voting-based detection (VD). To do so, we combine two detection mechanisms, and look at the gain of the overall performance.

Recall that Sybils could maximize the number of colluding votes by forming a tight complete-graph. So we first examine the detection performance over a graph G_n containing a completely-connected Sybil region with $N_s = 100$ and $N_{in} = 10$, the same configuration as Fig. 9. Fig. 13 shows the recall of the combined methods as the average number of attack-links varies from 0 to 1. We see that the community detection could promote recall to 100% even the voting-based detection just gains a recall of 2%. This means the community detection is able to find the whole Sybil community, once the voting-based detection is able to provide a number of Sybil seeds.

A possible way that the attacker makes the community detection harder is to weaken the collusion. To simulate this strategy, we remove half of links between the Sybils, and let $G_{n/2}$ represents the new graph. Fig. 13 shows that the community detection achieves the similar performance in the case of weak collusion. The reason is that Sybils get fewer positive votes after weakening the collusion. Thus, the voting-based detection achieves a relatively higher recall, which in turn feeds the community detection with more seeds and counteracts the effect of weak collusion.

Fig. 14 shows the precision of the Sybil community

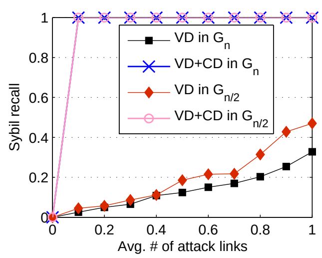


Fig. 13. Sybil recall with-/without community detection

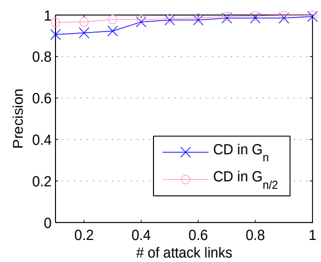
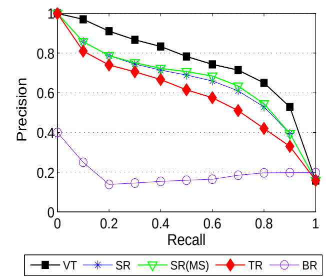
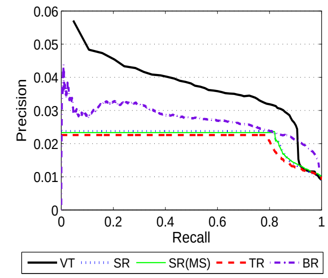


Fig. 14. The precision of Sybil community detection



(a) Manual checked dataset



(b) Banned dataset

Fig. 15. Recall precision curve for various Sybil detection schemes on different datasets. VT stands for VoteTrust; SR for SybilRank; SR(MS) for SybilRank with multi-community seeds; TR for TrustRank; and BR for BadRank.

detection, measured by the fraction of nodes that are simulated Sybils in the identified community. We see that the community detection achieves a nearly 100% precision as Sybils create more attack-links. Interestingly, the community detection is more accurate when the attacker weakens Sybil collusion. Because in that case, the community detection could get more Sybil seeds to rightly expand the Sybil community.

The combination of these results show that the community detection of VoteTrust could effectively detect other colluding Sybils, and thus preventing them from creating attack-links in a proactive manner.

6.3 Detecting Real Sybils

To show the advantage of VoteTrust, we compare it against typical ranking schemes that also use PageRank-like algorithm to propagate scores, including SybilRank (SR), TrustRank (TR) and BadRank (BR). The comparison actually shows *the extent to which VoteTrust can improve existing schemes by incorporating the negative links.*

Ground-truth datasets. We first run VoeTrust and other ranking algorithms over the PKU network to detect real Sybils, and then compare their performance using two ground-truth datasets: The first one contains 500 randomly selected PKU users. An expert team carefully scrutinized all accounts to classify them as real users or Sybils by looking over detailed profile data. In particular, they examine whether the accounts use photos of attractive young women/men, send or forward spam messages or links and use invalid email addresses. This manual inspection finds 73 Sybil accounts[2] in the dataset. We use this dataset to evaluate the performance of different defenses. The other contains 2502 PKU Sybil accounts that already detected by Renren security team using prior techniques. Since it is a more large ground truth data about Sybils, we use this dataset to confirm the effectiveness of VoteTrust in detecting true Sybils.

Seed selection. For a fair comparison, we strive to use the same trusted seeds for all schemes. We inspect 100 nodes of high inverse PageRank scores and only seed trust at the nodes passing the verification. For bad seeds, we select Sybils based on the number of their in-links, because the bad score propagates from the receiver to the sender (i.e., the reverse direction of trust score propagation). So we select 100 highest in-degree Sybils from the 2502 banned accounts as the bad seeds of BadRank.

Results. Fig. 15(a) and Fig. 15(b) plot the *precision-recall curve* (PRC) of VoteTrust and other approaches on two datasets, respectively. We see that VoteTrust outperforms all other schemes due to incorporating the negative link information. Given the same recall, VoteTrust achieves the highest precision on both datasets. It should be pointed out that the second dataset provided by Renren has a *limitation* that it does not contain the Sybils that the security team fails to find. With this limitation, we may significantly underestimate the true precision of different schemes. In particular, the precision is calculated based on the assumption that only banned accounts in the dataset are Sybils. Thus, we have to suppose that the system is wrong when it actually finds Sybils out of the dataset, which underestimates the true precision. This underestimation is the reason why the all schemes produce unreasonably low precision in Fig. 15(b). However, the underestimation does not affect the relative improvement of VoteTrust over others. Further, the horizontal line in Fig. 15(b) is because about 80% of the banned accounts have the same lowest score.

VoteTrust versus TrustRank. TrustRank also leverages the heuristic that Sybils have few in-links, and propagates the trust score from trusted seeds to other users. However, TrustRank may mix Sybils with many non-popular users that also have low trust score due to few incoming links. In contrast, VoteTrust further leverages the information of negative links to distinguish Sybils from non-popular users. On average, given a similar recall, VoteTrust can improve the precision of TrustRank by 32.9% in manually checked dataset, and by 50.1% when limiting the recall$\geq$ 80% in banned accounts dataset.

VoteTrust versus SybilRank. SybilRank [16] uses a early-terminated random walk to propagate trust score.

2. This fraction is higher than the whole networks. This is mainly because PKU is the most famous university in China, and many Sybils disguise as PKU users to increase their popularity.

Like [16], we perform $\log n$ power iterations for SybilRank, where n is the size of the graph. Fig. 15 shows that SybilRank outperforms TrustRank, but it performs worse than VoteTrust. For example, SybilRank has about 10% higher false positive rate than VoteTrust in the manually checked dataset, given a similar false negative rate. Similar to TrustRank, SybilRank also has no additional information to distinguish Sybils and non-popular users. In contrast, VoteTrust further leverages the sign of nodes' outgoing links (i.e., the positive/negative feedbacks), as Sybils have a higher percentage of negative outgoing links to real users than that of non-popular users.

We also implement the original seed selection strategy used in SybilRank, where seeds are picked from different communities. With the Louvain method, we found 116 communities in friend invitation graph, among which 50 large communities contain more than 94% of total nodes. We inspect 2 nodes in each community and designate as SybilRank trust seeds the nodes that pass the manual verification. From Fig. 15 we see that SybilRank with this original strategy performs only slightly better than that with inverse PageRank strategy, since nodes of high inverse PageRank scores usually have links to multiple communities in the friend invitation graph. SybilRank still performs worse than VoteTrust.

VoteTrust versus BadRank. BadRank is based on the premise that a node is Sybil if it points to another Sybil. Different from TrustRank, BadRank propagates the bad score from Sybil seeds to users who link to Sybils. However, The performance of BadRank is significantly depend on Sybil seeds and may punish innocent users that are enticed to send requests to Sybils. We compare the performance of BadRank and VoteTrust on both human checked samples and real banned accounts.

We find that BadRank cannot efficiently detect Sybils on human checked dataset, because many Sybils are not in the seed's community, and thus cannot be detected by BadRank. For banned accounts dataset, BadRank outperforms both TrustRank and SybilRank due to selecting seeds from these Sybil accounts. However, it has at least 30% higher false positive rate than VoteTrust on average, given a similar false negative rate.

6.4 Discussion

How to handle false rates in practice? In the real OSN, there are a wide variety of Sybils. Relying on a few abnormal features can be very difficult to yield a binary Sybil/non-Sybil classifier with both high recall and high precision. This is true for all the Sybil defenses so far. However, this limitation does not appear to significantly affect the ability of VoteTrust in practice:

First, we find that many Sybils are missed because they have not launched attacks yet (i.e., inactive actually). For example, in the ground-truth dataset used in Fig.15(a), about 25% Sybils have no behavioral data (e.g. friend requests) to leverage for classification. These accounts cannot do harm to real users until after they start generating friend requests to real users or their colluders. In this case, VoteTrust can recall these accounts as quickly as possible once they become active, to minimize the amount of damage they can do to real users.

Second, we find that most false positives in VoteTrust can be attributed to promiscuous real users that also get low acceptance rates due to friend request abuse. Thus, OSNs could tolerate more false positives for acceptable recall. Before banning a suspected account, OSN (like Renren) sends CAPTCHAs and other challenges to the account. In the case of false positives, OSN can regulate the abusive behavior of promiscuous users, since passing the challenges requires human effort. In the case of right detection, fake accounts cannot correctly pass the challenge and would be banned.

Does the experiment violate user privacy? Renren's privacy policy allows the users in the same affiliation to see the profile information and friend relationships of each other. Thus, the personal information contained in our PKU dataset is actually public to us given the same affiliation. Notice that we examine the personal information for obtaining ground-truth dataset. The VoteTrust itself does not require these personal information, the OSN provider can deploy the detector solely based on friend request records.

Is it possible to leverage promiscuous users to attack? Since promiscuous real users are open to befriending even strangers, the attacker may want to use well-maintained Sybils (e.g., good-looking female accounts) to gain positive links from these users. However, this targeted attacks is difficult when the attackers have limited knowledge about which users are gullible. Promiscuous users have high tolerance of unwanted communication, while cautious users are more resistant to Sybils. The attacker cannot distinguish these types of OSN users. As a result, Sybils are likely to receive many negative feedbacks from cautious users, although they may be able to contact some promiscuous users.

An alternative strategy of Sybils is to wait for receiving the requests from promiscuous users, without contacting them first. Notice that Sybil accounts are not popular in OSNs, so promiscuous users rarely encounter/contact Sybils first given a huge user population.

7 Real-world Deployment

With the help of supportive collaborators at Renren, we were able to deploy the VoteTrust system at the company for internal testing on the global graph.

VoteTrust system consists of Sybil detection and community detection components, which can be used in conjunction to defend against Sybil attacks. The Sybil detection component takes the friend invitation graph G and a set of trusted seeds as inputs, and outputs a set S of active Sybils that achieve the detection bound on the number of attack-links. Once Sybil nodes are

Trust-based votes assignment

```
compute(Msgs){
  vote = 0 or 1/N;
  for (m:Msgs) sum += m;
    vote = d*sum + (1-d)*I(u);
  if (superstep < 30)
    sendMsgToNbrs(vote/#neighbors);
  else
    voteTohalt();}
```

Global vote aggregating

```
compute(Msgs){
  rate = 0.5;
  for (m:Msgs) {
    sum+=m;
    if (sign(u,v) > 0)
       weights+=m;}
  rate = weights/sum;
  if (superstep < 30)
    sendMsgToNbrs(rate * vote);
  else
    voteTohalt();}
```

Fig. 16. The implementations of Sybil detection using Giraph.

Local community identification

```
compute(Msgs){
  belong = 1 or 0;
  for (m:Msgs)
     if m == 0
       out += sign(u,v);
     else
       in += sign(u,v);
  if (in - out > 0)
     belong = 1;
  if (superstep < 30)
    sendMsgToNbrs(belong);
  else
    voteTohalt();
  }
```

Fig. 17. The implementations of community detection using Giraph. We omit the implementation of bad-score assignment, since it is similar to that of trust-based votes assignment except for the seed set.

identified, the community detection algorithm detects other colluding Sybils around those identified Sybils.

7.1 Distributed Implementation

VoteTrust has inherent high parallelism, since a node's state only depends on its neighbors. Thus, we implement VoteTrust using the graph processing system, which enables us to process the entire graph that is very large. Specially, we implement VoteTrust on the Giraph [17], an open-source clone of Google's Pregel.

Like Pregel, the organization of Giraph programs is inspired by Bulk Synchronous Parallel model [18]. Typical Giraph computation consists of a sequence of iterations, called supersteps. During each superstep, Giraph invokes a user-defined function `compute()` for each vertex, conceptually in parallel. The function specifies behavior at a single vertex u and a single superstep S. It takes messages sent to u in superstep $S-1$ as input, modifies the state of u and sends output messages to other vertices that will be received at superstep $S+1$ through `sendMsgToNbrs()` function. The algorithm terminates when all vertices vote to halt using `voteTohalt()` function and there are no messages in transit.

Fig. 16 gives the implementation of the Sybil detection component. Taking the vote capacity assignment as an example, in each *superstep*, we implement the `compute()` function. It updates the node u's vote capacity based on Eq.(2), and calls `sendMsgToNbrs()` function to send the new capacity value to u's each neighbor. After a number of iterations, vertex calls `voteTohalt()` to terminate computation. Fig. 17 gives the implementation of the community detection component. In the `compute()` function, a node u first computes $\Omega(u)$. If $\Omega(u) < 0$, the node u would also be included in the Sybil community.

Notice that Giraph only maintains the outgoing neighbors of each node. To facilitate the implementation of VoteTrust, we separate the invitation graph into two graphs: i) **Link initiation graph:** a directed graph where a directed link (u, v) represents a node u sends a friend request to another node v. ii) **Link acceptance graph:** a weighted-directed graph where a directed link (u, v) represents u receives a request from v (the inverting direction of link initiation edge). The weight equals 1 (or -1) if u accepts (or rejects) the request. We run the first stage of VoteTrust (i.e., vote capacity assignment) on the link initiation graph, whereas running other stages on the link acceptance graph.

7.2 Real-world Detection Result

In October, 2013, we deploy Giraph system on Renren's computation cluster, and run our VoteTrust algorithm shown in Fig. 16 and Fig. 17. To get the implicit negative links, we replay the friend request log in recent two years (from 1-Oct-2011 to 1-Oct-2013). A pair of users has a negative link if the request that one sends to the other has not been accepted till the end of log.

Sybil Detection. VoteTrust has two thresholds: a node is consider as Sybil if its global acceptance rate is below a threshold δ_f, and a node's votes are ignored if its vote capacity is below a threshold of δ_v. The method of setting the second threshold is similar to that in Section 6.2.1.

To derive the first threshold, Renren performs manual inspection on different threshold choices $\Sigma = \{\tau, 2\tau, \ldots, 1\}$, where the granularity τ depends on the

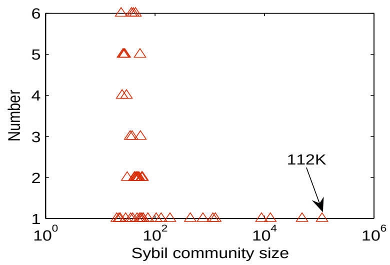


Fig. 18. The distribution of Sybil community size

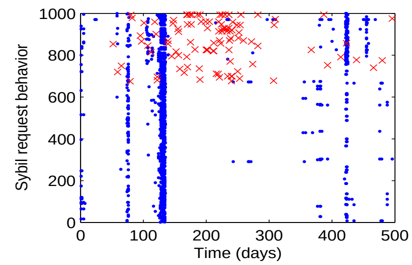


Fig. 19. The temporal behavior of 1000 Sybils selected from the largest community. The "•" (or "×") point represents the creation of collusion (or attack) link.

amount of resources for manual inspection. In particular, for each $\delta_f \in \Sigma$, the security team at Renren gets the set of suspected users whose global acceptance rates below δ_f (say V_f). The security team samples random users from V_f, and reports a portion of fakes (say α_f) after manual inspection. By this way, Renren could estimate the precision (indicated by α_f) and recall (indicated by $|V_f| \times \alpha_f$) for each threshold choice. With this estimation, Renren can select the threshold δ_f that balances precision and recall. Notice that the threshold could also be set automatically if the OSN operator already has ground-truth data to train the system. Let T and S represent the set of real users and Sybils in the ground-truth data. For any threshold choice $\delta_f \in \Sigma$, we can get the precision as $|V_f \cap S|/|V_f \cap (T \cup S)|$ and recall as $|V_f \cap S|/|S|$, and thus get the balanced threshold.

After setting the thresholds, Renren begin to send CAPTCHAs and other challenges to the suspected users whose global acceptance rates below the balanced threshold. If the suspected user cannot pass the challenges, the account is banned. Also, Renren finds suspected users whose global acceptance rates are much lower than their local acceptance rates (i.e., the percentage of their positive links), which indicates these suspected users have colluders. Then, Renren takes them as bad seeds and finds other colluders with the community detection component of VoteTrust.

Detection Results: As Renren have deployed an online Sybil detector using *local* acceptance rate [2], we prune Sybils whose local acceptance rates are already below the threshold, and focus on colluding Sybils that cannot be identified based on their local property. Given the two-year observation window, VoteTrust detects 105 Sybil communities that contain a total of 190,378 colluding Sybils. Fig. 18 gives the size distribution of these communities. We see that VoteTrust is able to detect a number of very large Sybil communities, e.g., the largest one contains 112,538 Sybils.

To examine how the large community is formed, we randomly select 1000 Sybils from the largest community, and plot their temporal behavior in Fig 19. We see that these Sybils survive a long time by making the attack stealthy. Before creating attack links, Sybils first create many collusion links among themselves (e.g., around day 120), in order to keep high acceptance rates. Moreover, a fraction of Sybils rarely attack, which are only used to vouch others. However, VoteTrust can prune collusion links by aggregating the *global* acceptance rate, and thus find the attacking Sybils. Further, VoteTrust can identify Sybil community around the attacking Sybils, thus find other colluding Sybils even they have not created attack links yet.

False Positive: To assess the false positive, we examine their feedback to Renren's customer support department. Renren operates a telephone number and e-mail address where customers can attempt to get banned accounts reinstated. Complaints are evaluated by a human operator, who determines if the account was banned erroneously. We use the complaint rate, measured as the number of complaints divided by the number of accounts banned, as an upper-bound on false positives. During the one-month period after Sybils are banned, the average complaint rate on these banned accounts is about 1.33%, which is extremely low. We further check the detection precision with a manual inspection of 1000 banned Sybils chosen at random. Our manual inspection shows that 96.35% are *confirmed* spammers due to containing spam (*e.g.*, blog spam or post spam).

Making detection adaptive. In the future, Renren would execute VoteTrust periodically to detect newly created Sybils. After the detection threshold δ_f has been bootstrapped, Renren can use an adaptive feedback scheme to dynamically tune the threshold on the fly. The adaptive feedback is drawn from the customer complaint rate to Renren's support department. For example, Renren can raise or lower the threshold to maintain an acceptable complaint rate.

8 RELATED WORK

Recently, there has been a great effort in defending Sybils (e.g., spammers) in OSNs. This section discusses the difference between VoteTrust and these studies.

Social-graph-based approaches: Some decentralized protocols OSNs [5], [8], [10] leverage the social graph structure to defend against Sybils. These techniques rely on the assumption that fakes can befriend only few real accounts. However, our measurement on Renren Sybils has shown this assumption is not valid [2]. Hence, existing network-based Sybil defenses are unlikely to succeed in today's OSNs. Given the limitation of relying solely on the social network structure, an attractive way to improve on these schemes is to give Sybil defense schemes additional information.

Recent work [14] proposes íntegro, a Sybil detection system that integrates user activities into graph structures. íntegro starts by predicting victim accounts from user-level activities. After that, it limits the trust score entering the Sybil region by weighting the social graph such that edges incident to predicted victims have much lower weights than others. But íntegro relies on the accuracy of victim prediction, which requires the information of many behavioral features. Moreover, as long as the victims' links have non-zero weights, Sybils could still increase their scores by befriending even more victims.

However, VoteTrust only focuses on the friend invitation behavior, and detects Sybils that get more rejections than acceptances from real users, irrespective of the number of victims (i.e., those who accept Sybils). Also, VoteTrust is compatible with many exiting defense schemes such as SybilRank and íntegro. We can use them as the first step of VoteTrust (i.e., vote assignment), and then use the next step of VoteTrust (i.e., vote aggregation) to leverage the feedbacks on links.

Reputation systems: In many P2P systems, reputation systems have received a significant amount of attention as a solution for mitigating the affects of malicious peers. In an important work, Cheng and Friedman [19] classify them as *symmetric* or *asymmetric* approaches, and prove formally that the symmetric reputation systems (such as EigenTrust [20]) are susceptible to Sybil attacks.

Different from EigenTrust, VoteTrust is an asymmetric system that has trusted nodes from which reputation values propagate. It further limits the Sybil attack by combining the implicit information of negative links with the graph structure. As demonstrated in the paper, VoteTrust can significantly outperform other asymmetric systems that rely solely on graph topology (such as TrustRank [15] or BadRank [21]).

Sybil community detection: Pervious works (e.g., [7], [10]) are designed for the unsigned graph, and cannot detect communities in a signed graph like friend invitation graph. This is why we propose the VoteTrust scheme. Our algorithm could leverage the implicit information of negative links, and thus is able to accurately identify the Sybil community.

Feature-based approaches: Sybils are created for profitable malicious activities, such as spamming, click-fraud, malware distribution, and identity fraud. Hence, many works [3], [22] analyze aberrant behavior or spam content to detect Sybil accounts. Meanwhile, some Sybil detection systems have recently been developed based on Bayesian filters and SVMs [23], [24]. However, these feature-based Sybil detection systems require training on large samples of ground-truth data.

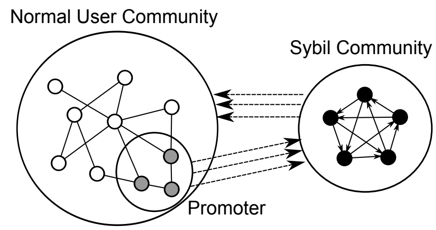


Fig. 20. Illustration of collusion-based attack. We call normal users (either intentionally or accidentally) sending requests to Sybils as promoters.

9 Conclusion and Limitation

This paper presents VoteTrust, a system that leverages user interactions of initiating and accepting links to defend against Sybil attacks. We provide the security guarantees of VoteTrust, demonstrating that we limit the number of requests Sybils can send to real users. Our evaluation over real network shows that VoteTrust is able to detect real Sybils with high precision, and significantly outperforms traditional ranking systems. Finally, working closely with Renren security team, we have deployed VoteTrust system at Renren, showing that VoteTrust can accurately detect real, large-scale Sybil collusion existing in the network.

Although we also use some standard techniques (e.g., a PageRank-style algorithm to propagate scores), we make three notable contributions: First, we introduce a *new graph model* for Sybil defense, which nicely combine link structure and user feedback. Second, we propose *new techniques*, including global vote aggregation and local community expansion, to exploit the negative links. Finally, we present and analyze theoretically the *security guarantees* of VoteTrust.

Limitation: Recent works [25], [26] have found that miscreants start to sell legitimate accounts that have been compromised in Twitter. Thus, if attackers are willing to buy friends from miscreants, they could enhance the attack capacity by increasing the number of in-links or the acceptance rate of Sybils. In this case, VoteTrust can be considered as a first-level defense that increases the attack cost, and could be combined with other orthogonal techniques (e.g., those proposed in [25], [26]) for even better defense.

Appendix A
Proof of Theorem 1

Proof: We now provide detailed proofs of the VoteTrust's security properties. An typical attack mode is illustrated in Fig. 20. The attacker would connect Sybils into a complete-connected graph, so that each Sybil could receive the maximum number of collusion votes from others. Let N_{in} be the in-links going from the non-Sybil region to the the Sybil region.

According to the work [27], the total vote capacity that flows into the Sybil community, say E_S, is given by,

$$E_S = E_S^{in} - E_S^{out} \tag{10}$$

This equation means that the total vote capacity of a Sybil community depends on the difference between incoming capacity E_S^{in} and outgoing vote capacity E_S^{out}. And E_S^{in} and E_S^{out} can be calculated using the following expressions,

$$E_S^{in} = \frac{d}{1-d}\sum_{i\in V_{in}} f_i x_i \tag{11}$$

$$E_S^{out} = \frac{d}{1-d}\sum_{i\in V_{out}} (1-f_i) x_i \tag{12}$$

where V_{in} represents the set of normal users that link to Sybils, and V_{out} is the set of Sybils that link to normals. f_i represents the fraction of a node i's edges that link to Sybils. d is the decay factor in Eq.(1), and we use α to denote $d/(1-d)$.

Let $\bar{x}_l$ and $\bar{x}_s$ represent the average vote capacity for each node in non-Sybil region and Sybil region, respectively. To be fully connected, each Sybil sends $N_S/2$ requests to others Sybils. Here, N_s is the size of Sybil set. Thus, for $i \in V_{out}$, $1 - f_i = \frac{N_{out}}{N_{out}+N_S/2}$. We denote $f_{in} = E[f_i]$ for $i \in V_{in}$. Substituting it into Eq.(10), we get,

$$E_S = \alpha\left(N_{in} f_{in}\bar{x}_l - E_S\frac{N_{out}}{N_{out}+N_s/2}\right) \tag{13}$$

Solving the above equation, the total vote capacity of Sybil community is,

$$E_S = \frac{\alpha N_{in} f_{in}\bar{x}_l}{\frac{\alpha N_{out}}{N_{out}+N_S/2}+1}. \tag{14}$$

Due to Sybils have similar structure, we assume they have equal global acceptance rate $\bar{p}_s$. Let $\bar{p}_l$ be the average global acceptance rate of promoters. According to the rating model Eq.(2), we have,

$$\bar{p}_s = \frac{\frac{N_s}{2}\bar{x}_s\bar{p}_s + \gamma N_{out}\bar{x}_l\bar{p}_l}{\frac{N_s}{2}\bar{x}_s\bar{p}_s + N_{out}\bar{x}_l\bar{p}_l} \tag{15}$$

where γ is the acceptance percentage of Sybils' out-links. Based on Eq.(15), we can get the Sybils' total out-links as:

$$N_{out} = \frac{\frac{N_s\bar{x}_s}{2}(\bar{p}_s - \bar{p}_s^2)}{\bar{x}_l\bar{p}_l(\bar{p}_s-\gamma)} \tag{16}$$

Notice that $\bar{p}_s \in [0,1]$, we get $\frac{\partial N_{out}}{\partial \bar{p}_s} < 0$. This indicates that N_{out} decrease as $\bar{p}_s$ grows.

To evade the detection, Sybils should maintain their global rating $\bar{p}_s$ above the detection threshold δ_f. Therefore, N_{out} has the least upper bound when $\bar{p}_s = \delta_f$. Note that $E_S = N_s\bar{x}_s$, substituting Eq.(14) into Eq.(16), and let $\bar{p}_s = \delta_f$ yields the inequality,

$$(4\alpha+4)aN_{out}^2 + (2aN_s - 2bc)N_{out} - bcN_s \le 0 \tag{17}$$

where $a = \delta_f - \gamma$, $b = \delta_f - \delta_f^2$ and $c = \frac{\alpha N_{in} f_{in}}{p_l}$.

According to the property of quadratic function, we know the condition that the above inequality holds is $N_{out} \ge 0$. Thus, we get the following inequality:

$$\begin{aligned} N_{out} &\le \frac{2bc - 2aN_s + \sqrt{(2aN_s-2bc)^2 + (16\alpha+16)abcN_s}}{8a\alpha+8a} \\ &= \frac{2bc - 2aN_s + 2\sqrt{(aN_s+(2\alpha+1)bc)^2 - (4\alpha^2+4\alpha)b^2c^2}}{8a\alpha+8a} \\ &\le \frac{2bc - 2aN_s + 2aN_s + (4\alpha+2)bc}{8a\alpha+8a} \\ &= \frac{bc}{2a} \end{aligned}$$

Substitute the expressions of a, b and c into the above inequality, and let $\rho = \frac{\alpha f_{in}}{2\bar{p}_l}$, we get the upper bound given in equation (4). Theorem 1 is proved.

In the theorem, the constant factor $\rho = \alpha f_{in}/2\bar{p}_l$. where $\alpha = d/(1-d)$ and d is the decay factor in Eq.(1). f_{in} represents the ratio of the number of in-links of Sybils to the total out-degree of promoters. $\bar{p}_l$ indicates the average global acceptance rate of promoters. □

Appendix B
Proof of Theorem 2

Proof: In a completely connected Sybil group with size N_s, each Sybil gets $\bar{x}_l$ votes on average. Thus the total vote capacity in this community is $E_S = N_s\bar{x}_s$. Substituting it into Eq.(14) we get,

$$N_s\bar{x}_s = \frac{\alpha N_{in} f_{in}\bar{x}_l}{\frac{\alpha N_{out}}{N_{out}+N_s/2}+1} \tag{18}$$

Note that the votes will be ignored by VoteTrust when $\bar{x}_s < \delta_v$, so N_s has the lowest upper bound when $\bar{x}_s = \delta_v$. Substituting this equation into Eq.(18), we get

$$\frac{\delta_v}{2}N_s^2 + \left(a\delta_v - \frac{bN_{in}}{2}\right)N_s - bN_{in}N_{out} \le 0 \tag{19}$$

where $a = (\alpha+1)N_{out}$ and $b = \alpha f_{in}\bar{x}_l$ both defined in Theorem 1. Solving the inequality like that of inequality (17) yields the upper bound of community size in Eq.(5). Theorem 2 is proved. □

Appendix C
Proof of Theorem 3

Proof: The bad score assignment is similar as the vote capacity assignment in Fig 20. The difference is that it propagates the bad score along the reverse direction of links from Sybil seeds.

According to the work [27], the total bad score (denoted as E_N) in normal non-Sybil region (see Fig. 20) depends on the difference between incoming bad score E_N^{in} and outgoing bad score E_N^{out}.

$$E_N = E_N^{in} - E_N^{out} \tag{20}$$

Due to the essential process of bad score assignment is equivalent to the trust-based vote assignment (in

Theorem 1), the total bad score of normal users receive from bad seeds can be derived based on Eq.(14),

$$E_N = \frac{\alpha f_{in} N_{in} \bar{x}_s}{\frac{\alpha N_{out}}{N_{out}+N/2} + 1}. \quad (21)$$

where, α and f_{in} are constant parameters. Similarly, N_{out} is the number of attack links, and $\bar{x}_s$ is the average bad score of Sybils.

Straightforwardly, the number of normal users whose bad score are higher than Sybils' average score does not exceed $\frac{E_N}{\bar{x}_s}$, which is,

$$|\{u_i | x_i > \bar{x}_s\}| < \frac{\alpha f_{in} N_{in}}{\frac{\alpha N_{out}}{N_{out}+N/2} + 1} < \alpha f_{in} N_{in} \quad (22)$$

Theorem 3 is proved. □

References

[1] J. R. Douceur, "The Sybil attack," in *Proc. of IPTPS*, March 2002.

[2] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai, "Uncovering social network sybils in the wild," in *Proc. of IMC*, 2011.

[3] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao, "Detecting and characterizing social spam campaigns," in *Proc. of IMC*, 2010.

[4] C. Grier, K. Thomas, V. Paxson, and M. Zhang, "@spam: the underground on 140 characters or less," in *Proc. of CCS*, 2010.

[5] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: defending against sybil attacks via social networks," in *Proc. of SIGCOMM*, 2006.

[6] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against sybil attacks," in *Proc. of IEEE S&P*, 2008.

[7] W. Wei, F. Xu, C. C. Tan, and Q. Li, "Sybildefender: Defend against sybil attacks in large social networks," in *Proc. of INFOCOM*, 2012.

[8] G. Danezis and P. Mittal, "Sybilinfer: Detecting sybil nodes using social networks," in *Proc of NDSS*, 2009.

[9] N. Tran, B. Min, J. Li, and L. Subramanian, "Sybil-resilient online content voting," in *Proc. of NSDI*, 2009.

[10] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove, "An analysis of social network-based sybil defenses," in *Proc. of SIGCOMM*, 2010.

[11] J. Xue, Z. Yang, X. Yang, X. Wang, L. Chen, and Y. Dai, "Votetrust: Leveraging friend invitation graph to defend against social network sybils," in *Proc. of INFOCOM*, 2013.

[12] J. Jiang, C. Wilson, X. Wang, P. Huang, W. Sha, Y. Dai, and B. Y. Zhao, "Understanding latent interactions in online social networks," in *Proc. of IMC*, 2010.

[13] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda, "All your contacts are belong to us: automated identity theft attacks on social networks," in *Proc. of WWW*, 2009.

[14] Y. Boshmaf, D. Logothetisy, G. Siganosz, J. L. Jorge Leríax, M. Ripeanu, and K. Beznosov, "Íntegro: Leveraging victim prediction for robust fake account detection in osns," in *Proc. of NDSS*, 2015.

[15] Z. Gyongyi, H. Garcia-molina, and J. Pedersen, "Combating web spam with trustrank," in *VLDB*. Morgan Kaufmann, 2004, pp. 576–587.

[16] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, "Aiding the detection of fake accounts in large scale social online services," in *nsdi*, 2012.

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

[18] L. G. Valiant, "A bridging model for parallel computation," *Commun. ACM*, vol. 33, no. 8, pp. 103–111, Aug. 1990.

[19] A. Cheng and E. Friedman, "Sybilproof reputation mechanisms," in *Proc. of P2PECON*, 2005.

[20] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in P2P networks," in *Proc. of WWW*, Budapest, Hungary, May 2003.

[21] S. Ghosh, B. Viswanath, and F. K. et al, "Understanding and combating link farming in the twitter social network," in *Proc. of WWW*, 2012.

[22] K. Thomas, C. Grier, D. Song, and V. Paxson, "Suspended accounts in retrospect: an analysis of twitter spam," in *Proc of IMC '11*, New York, NY, USA, 2011, pp. 243–258.

[23] A. H. Wang, "Don't follow me: Spam detection on twitter," in *Proc. of SECRYPT*, Athens, Greece, July 2010.

[24] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proc. of ACSAC*, Austin, TX, December 2010.

[25] G. Stringhini, G. Wang, M. Egele, C. Kruegel, G. Vigna, H. Zheng, and B. Zhao, "Follow the green: Growth and dynamics in twitter follower markets," in *ACM Internet Measurement Conference (IMC)*, 2013.

[26] G. Stringhini, M. Egele, C. Kruegel, and G. Vigna, "Poultry Markets: On the Underground Economy of Twitter Followers," in *Proceedings of the Workshop on Online Social Network (WOSN)*. Helsinki, Finland: ACM, August 2012.

[27] M. Bianchini, M. Gori, and F. Scarselli, "Inside PageRank," *ACM Trans. Internet Technol.*, vol. 5, no. 1, pp. 92–128, Feb. 2005.

PLACE PHOTO HERE

Zhi Yang received the BS degree from Harbin Institute of Technology in 2005 and the PhD degree in computer science from Peking University in 2010. He is currently an associate professor at Peking University. His research Interests include the areas of security and privacy, networked and distributed systems, and data intensive computing. Most recently, he studying spam issues in online social networks, as well as designing systems for processing large graphs.

PLACE PHOTO HERE

Jilong Xue received the BS degree from Xidian University in 2009. He is currently a PhD student in Peking University. His research Interests include the areas of security and privacy, networked and distributed systems.

PLACE PHOTO HERE

Xiaoyong Yang received the BS degree from Beijing University of Post and Telecommunication in 2010 and the MS degree from Peking University in 2013. He is currently the engineer of security team of Renren.

PLACE PHOTO HERE

Xiao Wang received the BS and MS degrees in computer science from Peking University in 2009 and 2012, respectively. He is currently the director of system and security teams of Renren.

PLACE PHOTO HERE

Yafei Dai received the PhD degree in computer science at Harbin Institute of Technology. She is currently a professor at the Computer Science Department, Peking University. Her research areas include networked and distributed systems, P2P computing, network storage, and online social networks. She is a member of the IEEE.