SocialFilter: Introducing Social Trust to Collaborative Spam Mitigation

Michael Sirivianos
Telefonica Research
msirivi@tid.es

Kyungbaek Kim
UC Irvine
kyungbak@uci.edu

Xiaowei Yang
Duke University
xwy@cs.duke.edu

1 Introduction

Centralized email reputation services that rely on a small number of trusted nodes to detect and report spammers, e.g., [1, 5, 6], are being challenged by the increasing scale and sophistication of botnets. Moreover, several of these services require paid subscription (e.g., CloudMark [1] and TrustedSource [5].)

Motivated by the shortcomings in terms of effectiveness and cost of email reputation services, researchers have proposed open and collaborative peer-to-peer spam filtering platforms, e.g., [29, 30]. These collaborative systems assume compliant behavior from all participating spam reporting nodes, i.e., that nodes submit truthful reports regarding spammers. However, this is often an unrealistic assumption given the fact that these nodes may belong to distinct trust domains.

To address the above challenges, we propose SocialFilter: a collaborative spam filtering system that uses social trust embedded in Online Social Networks (OSN) to assess the trustworthiness of spam reporters. SocialFilter aims at aggregating the experiences of multiple spam detectors, thus democratizing spam mitigation. It is a trust layer that exports a measure of the system's belief that a host is spamming. Thus, it enables nodes with no spam detection capability to collect the experiences of nodes with such capability and use them to classify email connection requests from unknown email senders.

Each SocialFilter node submits *spammer reports* (§2.2) to a centralized repository. These reports concern spamming Internet hosts identified by their IP addresses. The goal of the system is to ensure that the reports reach other nodes prior to spamming hosts contacting those nodes, and that the spammer reports are sufficiently credible to warrant action by their receivers.

SocialFilter nodes are administered by human administrators (admins). Our insight is that nodes maintained by trusted admins are likely to generate trustworthy spammer reports. The repository utilizes a trust inference method that leverages trust transitivity to assign to each node a *reporter trust* (§3.1) value. This value reflects the system's belief that the spammer reports of a node are reliable.

However, transitive trust schemes are known to be vulnerable to the Sybil attack [9, 10]. To mitigate this attack, we use the social network to assess the belief that a node is a Sybil attacker, which we refer to as *identity uniqueness* (§3.2). Each node is associated with its administrator's identity. The identity uniqueness of a node is determined via the social network of administrators using a SybilLimit-based technique [27].

The originator of a spammer report assigns a confidence level to its report. The reporter trust of a node, its identity uniqueness and its confidence level determine how much weight the repository should place on its report. Subsequently, the repository combines the reports to compute a *spammer belief* value for each reported host IP. This value is exported by the repository to online systems that are tasked with blocking spam.

A recent unwanted traffic mitigation system, 0stra [18], combats unwanted traffic by forcing it to traverse social links annotated by credit balances. The per-link credit balances rate-limit unwanted communication. Unlike Ostra, SocialFilter does not use social links to rate-limit unwanted traffic. Instead it utilizes social links to bootstrap trust between reporters, and to suppress Sybil attacks. Our secondary contribution lies in the comparison between Ostra's and SocialFilter's approach in leveraging social trust. Ostra uses the social network as a rate-limiting conduit for communication. SocialFilter on the other hand uses the social network as a trust layer from which information on the trustworthiness of spam detectors can be extracted.

We demonstrate through simulation that collaborating SocialFilter nodes are able to suppress spam email traffic in a reliable and responsive manner. Our comparison with Ostra shows that our approach is less effective in suppressing spam when the portion of spammers in the network exceeds 1% and when spammers employ more than 100 Sybils each. However, Ostra can result in a non-negligible percentage of legitimate emails being blocked (false positives,) which is highly undesirable. In contrast, in this case SocialFilter yields substantially less false positives. Given the severity of false positives, these results suggest that our system can be a better alternative.

To the best of our knowledge, SocialFilter is the first OSN-based collaborative spam filtering system to use Sybil-resilient trust inference to assess the overall trustworthiness of a node's spammer reports. Our work demonstrates the plausibility of using social trust to improve the reliability and attack-resilience of collaborative spam mitigation systems.

2 System Overview

2.1 SocialFilter Components

Figure 1 depicts SocialFilter's architecture. The SocialFilter system comprises the following components: 1) human users that administer networked devices/networks (*admins*), join a social network and maintain a unique account; 2) *SocialFilter nodes* (or reporters) that are administered by specific admins and participate in monitoring and reporting the behavior of email senders; 3) *spammer reports* submitted by SocialFilter nodes concerning email senders they observe; and 4) a centralized repository that receives and stores spammer reports, and computes trust values.

The same admin that administers a SocialFilter node

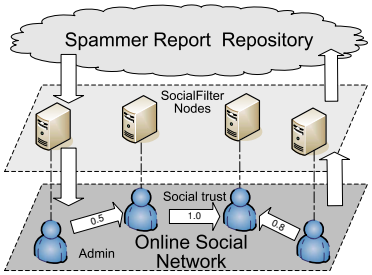


Figure 1: SocialFilter architecture.

also administers a group of online systems that interface with the node to report spamming behavior. Interfacing systems can be SMTP servers or IDS systems [19] that register with the SocialFilter repository. The interfacing system can also be one driven by a human user who reports an email (and consequently its originating email server) as spam.

2.2 Spammer Reports

An email characterization application uses the `ReportSpammer(`h`, confidence)` call of the SocialFilter node RPC API to feedback the observed behavior of an email sender h to the node. The first argument identifies the email sender, i.e., an IP address. The second argument is the confidence with which the application is reporting that the specified host is a spammer. The confidence takes values from 0% to 100% and reflects the fact that in many occasions traffic classification has a level of uncertainty. For example, an email server that sends both spam and legitimate email may or may not be a spammer. For instance, the confidence may be equal to the portion of emails received by host h that are spam [21].

Subsequently, the node submits a corresponding spammer report to the repository to share its experience. For example, if a node i's spam analysis indicates that half of the emails received from host with IP h are spam, i reports:

$$[\text{spammer report}]\ h,\ 50\%$$

To prevent forgery of reports and maintain accountability, nodes authenticate with both the repository and the OSN provider using standard single-sign-on authentication techniques, e.g., Facebook Connect [4].

2.3 Determining whether a Host is Spamming

Our system relies on the fact that nodes comprising Internet systems such as email servers, honeypots, IDS, etc, are administered by human admins. These users maintain accounts in online social networks (OSN.) The SocialFilter centralized repository utilizes two dimensions of trust embedded in OSNs to determine the trustworthiness of the reports submitted by SocialFilter nodes:

Reporter trust. The SocialFilter repository computes *reporter trust* values for all nodes by employing a transitive trust inference mechanism. This mechanism compares the reports of SocialFilter nodes that are socially acquainted to derive pairwise *direct trust* values (§3.1). If two friend nodes i and j have submitted reports concerning the same hosts, the repository can compare their reports to determine the direct trust value d_{ij}. The repository initializes the direct trust d_{ij} to a trust value explicitly submitted by the admin of i. This value is i's assessment on his friend's j ability to correctly maintain its SocialFilter node.

Identity uniqueness. The repository defends against Sybil attacks [10] by exploiting the fact that OSNs can be used for resource testing [18, 24, 27]. The test in question is a Sybil attacker's ability to create and sustain acquaintances. Using a SybilLimit-based [27] technique (§3.2,) the OSN provider assigns an *identity uniqueness* value to each node. This value quantifies the system's belief in that node not being a Sybil.

An application can use the `IsSpammer(h)` call of the SocialFilter node RPC API to obtain a *spammer belief* value, which quantifies how certain the system is that host h is spamming. The node obtains this value by querying the repository. The repository derives this value by aggregating spammer reports concerning h, and these reports are weighted by the reporter trust and identity uniqueness of the nodes that submitted them.

2.4 Assumptions

In designing SocialFilter, we make the following assumptions. We assume that competent and trustworthy SocialFilter admins have correctly configured their spam detection systems, so that their SocialFilter node sends mostly correct reports. We also assume that when they report the same spamming host, their reports mostly match, since a host is expected to send spam to most of the nodes it connects to [26]. In the rest of this paper, we call correctly configured and trustworthy SocialFilter reporters *honest*.

We also assume that the OSN provider and the SocialFilter repository reliably maintain the social graph, and the spammer reports. We trust the repository to correctly compute the spammer belief values.

Last, we assume that social connections between admins have been properly vetted, i.e., when two admins connect to each other in the OSN this implies that they know each other and their ability to correctly maintain their systems.

2.5 Threat Model

SocialFilter is a collaborative platform aiming at suppressing malicious traffic. In addition, it is an open system, meaning that any admin with a social network account and a device can join. As such, it is reasonable to assume that SocialFilter itself will be targeted in order to disrupt its operation.

Malicious nodes may issue false reports aiming at reducing the system's ability to detect spam or at disrupting legitimate email traffic.

In addition, an adversary may attempt to create multiple SocialFilter identities aiming at increasing its ability to subvert the system using false spammer reports and direct trust updates. Defending against Sybil attacks without a trusted central authority that issues verified identities is hard. Many decentralized systems try to cope with Sybil attacks by binding an identity to an IP address. However, malicious users can readily harvest IP

addresses through BGP hijacking [20] or by commanding a large botnet. For more details on the threat model, please see [23].

3 Design

3.1 Reporter Trust

Malicious nodes may issue false spammer reports to manipulate the perceived belief that a host is a spammer. In addition, misconfigured nodes may also issue erroneous spammer reports. SocialFilter can mitigate the negative impact of malicious or incorrect reports by assigning higher weights to reports obtained from more nodes with higher reporter trust.

The repository maintains a reporter trust value $0 \leq rt_i \leq 1$ for each node i managed by an admin in the social graph. This trust value corresponds to the repository's estimation of the belief that node i's reports are accurate. It is obtained from three sources: a) manual trust assignments between friends in the social networks; b) spammer report comparison; and c) transitive trust.

To derive trust values, the repository needs to maintain the social graph $\mathcal{S}(\mathcal{V}, \mathcal{E})$ of the admins in the SocialFilter system. $\mathcal{V}$ denotes the set of the admins and $\mathcal{E}$ denotes the set of the friend connections between socially acquainted admins. The repository also maintains a reporter trust graph $T(\mathcal{V}, E)$. The vertices of this graph is the set of all SocialFilter admins as is the case for graph $\mathcal{S}(\mathcal{V}, \mathcal{E})$. The edges E are the edges in $\mathcal{E}$ annotated with *direct trust* values between acquainted SocialFilter nodes. Next, we describe how the direct trust values are derived and how the reporter trust values are computed using $T(\mathcal{V}, E)$.

User-defined trust. First, to initialize the direct trust values, the repository relies on the fact that nodes are administered by human users. Admins that are socially acquainted can assess each other's competence. An admin i tags his acquaintance admin j with a *user-defined trust* value $0 \leq ut_{ij} \leq 1$ based on his belief on j's ability to correctly configure his node. The repository uses this value to initialize the direct trust value between friend nodes i and j: $d_{ij} = ut_{ij}$. Users frequently use the OSN to add friends and to communicate with each other, thus the requirement for administrators to rate each other should not induce a substantial usability burden.

Spammer reports comparison. Second, the repository dynamically updates the direct trust d_{ij} by comparing spammer reports submitted by two friend nodes i and j. The spammer reports of two friend nodes i and j can be compared if both nodes have reported on the same host h. Intuitively, if i and j share similar opinions on h, i should place high trust in j's reports. Let $0 \leq v_{ij}^k \leq 1$ be a measure of similarity between i and j's k_{th} report on a common host. The repository updates i's direct trust to j using an exponential moving average:

$$d_{ij}^{k+1} = \alpha * d_{ij}^k + (1 - \alpha) * v_{ij}^{k+1} \quad (1)$$

As i and j submit more common reports, the direct trust d_{ij}^k gradually converges to the similarity of reports from i and j. α is a system parameter that affects the influence of history on direct trust assessment.

Transitive trust. Third, the repository incorporates direct trust and transitive trust [12, 13] to obtain the reporter trust value for i: rt_i. It does so by analyzing the reporter trust graph $T(\mathcal{V}, E)$ from the point of view of a small set of pre-trusted nodes in $\mathcal{V}$. These pre-trusted nodes are administered by competent admins that are fully trusted by the SocialFilter repository.

We use transitive trust for the following reasons: a) due to the large number of nodes, the admin of a pre-trusted node i cannot assign a user-defined trust ut_{ij} to every admin of a node j, as he may not know him; b) due to the large number of email-sending hosts, a pre-trusted node i may not have encountered the same hosts with another node j, thus the repository may be unable to directly verify j's reports; and c) even if a pre-trusted node i has a direct trust value for another node j, the repository can improve the correctness of rt_j by learning the opinions of other SocialFilter nodes about j.

The overall reporter trust rt_j can be obtained as the maximum trust path between a pre-trusted node i and the node j in the trust graph $T(\mathcal{V}, E)$. That is, for each path $p \in P$, where P is the set of all paths between nodes the pre-trusted node and j:

$$rt_j = max_{p \in P}(\Pi_{u \rightarrow v \in p} d_{uv}) \quad (2)$$

The above trust value is computed from the point of view of a single pre-trusted node. We repeat this process for every pre-trusted node. We then average the reporter trust values for all pre-trusted nodes to derive a final rt_j value. We use multiple pre-trusted nodes to ensure that there is a trust path from a pre-trusted node to most honest nodes j. We also use many pre-trusted nodes to limit the influence of attackers that manage to establish a high trust path with one of the pre-trusted nodes.

Similar to Credence [25], we use the maximum trust path because it can be efficiently computed with Dijkstra's shortest path algorithm in $O(|E| \log |\mathcal{V}|)$ time for the sparse social graph $T(\mathcal{V}, E)$. In addition, it yields larger trust values than the minimum or average trust path, resulting in faster convergence to high confidence on whether a host is spamming. Finally, it mitigates the effect of malicious nodes that have low direct trust value towards honest nodes.

We compute the reporter trust from the point of view of a few pre-trusted nodes, instead of the point of view of each node for two reasons: a) we would need to compute the maximum trust path from each of the hundrends of thousands of nodes in the social graph, which would result in a significant computation overhead; b) the system aims at assessing a ground truth fact (whether a host is a spammer) and not a subjective fact, therefore it is appropriate to incorporate the transitive trust from multiple points of view. We compute the maximum trust path from the pre-trusted nodes to all other nodes periodically to reflect changes in direct trust values.

3.2 Identity Uniqueness

Each node that participates in SocialFilter is administered by human users that have accounts with OSN providers. The system needs to ensure that each user's

social network identity is closely coupled with its SocialFilter node. To this end, SocialFilter employs single sign-on authentication mechanisms, such as Facebook Connect [4], to associate the OSN account with the spammer repository account.

However, when malicious users create numerous fake OSN accounts, SocialFilter's spammer belief measure can be subverted. Specifically, a malicious user a with high reporter trust may create Sybils and assign high direct trust to them. As a result, all the Sybils of the attacker would gain high reporter trust. The Sybils can then submit reports that greatly affect the spammer belief values.

We leverage existing OSN repositories for Sybil user detection. Using a SybilLimit-like [27] technique, OSNs can approximate the belief that a node's identity is not a Sybil. We refer to this belief as *identity uniqueness*.

Social-network-based Sybil detection takes advantage of the fact that most OSN users have a one-to-one correspondence between their social network identities and their real-world identities. Malicious users can create many identities or connect to many other malicious users, but they can establish only a limited number of trust relationships with real users. Thus, clusters of Sybil attackers are likely to connect to the rest of the social network with a disproportionately small number of edges, forming small quotient cuts.

SocialFilter adapts the SybilLimit algorithm to determine an identity-uniqueness value $0 \leq id_i \leq 1$ for each node i. This value indicates the belief that the administrator of node i corresponds to a unique user in real life and thus is not part of a network of Sybils. To be Sybil-resistant, SocialFilter multiplies the identity-uniqueness value id_i by the reporter trust to obtain the trustworthiness of node i's spammer reports. We describe in detail how we compute id_i in [23].

3.3 Spammer Belief

We now describe how we combine reporter trust, identity uniqueness and spammer reports to derive a the *spammer belief* score. We define *spammer belief* as a score in 0% to 100% that can be interpreted as the belief that a host is spamming: a host with 0% spammer belief is very unlikely to be a spammer, whereas a host with 100% spammer belief is very likely to be one.

A node i may have email classification functionality through systems that interface with it using the i's `ReportSpammer()` API. In this case, i considers only the reports of those systems in calculating the spammer belief. Node i uses the average (possibly weighted) confidence of those reports to compute the similarity of its reports with the reports of its friends, which is used to derive direct trust values.

At initialization time, SocialFilter nodes consider all hosts to be legitimate. As nodes receive emails from hosts, they update their confidence (§ 2.2). For efficiency, nodes send spammer report to the repository only when the difference between the previous confidence in the node being a spammer and the new confidence exceeds a predetermined threshold δ.

When a node i receives a new spammer report for h, this new report preempts an older report, which is thereafter ignored. Each spammer report carries a timestamp. The time interval during which a spammer report is valid and taken into account is a tunable system parameter.

A node i that does not have email classification functionality may receive multiple spammer reports originating from multiple nodes $j \in V_i$ and concerning the same host h. Subsequently, i needs to aggregate the spammer reports to determine an overall belief `IsSpammer(h)` that h is a spammer. Node i derives the spammer belief by weighing the spammer reports' confidence with the reporter trust and identity uniqueness of their reporters:

$$IsSpammer(h) = \frac{\Sigma_{j \in V_i^h} \, rt_j \, id_j \, c_j(h)}{S} Logistic(S) \tag{3}$$

In the above equation, $V_i^h \subseteq V_i \setminus i$ is the set of members in i's view that have posted a spammer report for h. In addition, $S = \Sigma_{j \in V_i^h} \, rt_j \, id_j$.

The factor $0 \leq Logistic(S) \leq 1$ discounts the belief in a host h being spammer in case the reporter trust and identity uniqueness of the nodes that sent a spammer report for h is low. It is used to differentiate between the cases in which there are only a few reports from non-highly trustworthy nodes and the cases there are sufficiently many and trustworthy reports. When S is sufficiently large, we should consider the weighted average of the confidence in the reports to better approximate the belief that a host is spammer. But when S is small we cannot use the spammer reports to derive a reliable spammer belief value. Based on these observations, we define the function $Logistic$ as the logistic (S-shaped) function of S, where b is a small constant set to 5 in our design:

$$Logistic(S) = \frac{1}{1 + e^{(b - 5S)}} \tag{4}$$

3.4 Centralized Repository

Practice has shown that centralized email infrastructures such as web mail providers and email reputation services can scale to millions of clients. Thus, to simplify the design and provide better consistency and availability assurances we use a centralized repository. This repository can comprise a well-provisioned cluster of machines or even a data-center.

When a node queries the repository for the spammer belief of a host, the repository is interested on the reports for a single host. These reports are sent by multiple nodes, thus for efficiency it is reasonable to index(key) the reports based on the hash of the host's IP.

4 Evaluation

We evaluate SocialFilter's ability to block spam traffic and compare it to Ostra [18]. The goal of our evaluation is two-fold: a) to illustrate the importance of our design choice, i.e., incorporating identity uniqueness; and b) to shed light on the benefits and drawbacks of SocialFilter's and Ostra's approach in using social links to mitigate spam. For a brief description of Ostra and an efficacy

study of performing SocialFilter's trust computations at the centralized repository, we refer the reader to [23],

4.1 Simulation Settings

For a more realistic evaluation, we use a $50K$-user crawled sample of the Facebook social graph [11]. The sample is a connected component obtained from a $50M$-user sample via the "forest fire" sampling method [17].. Each user in the social network is the admin of an email server, which we also refer to as a SocialFilter or Ostra node. Nodes can send and receive email connections.

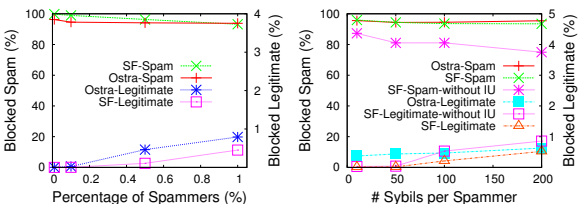
We have two types of nodes: *honest* and *spammers*. Honest nodes send 3 legitimate emails per day. 80% and 13% of the legitimate emails are sent to sender's friends and sender's friends of friends respectively, and the destination of the rest 7% emails is randomly chosen by the sender. Spammers send 500 spam emails per 24h, each to random honest nodes in the network. We set Ostra's credit bounds as $L = -5$ and $U = 5$. The above settings are obtained from Ostra's evaluation [18]. Honest and spammer nodes correspond to users uniformly randomly distributed over the social network.

Several nodes can instantly classify spam connections. These instant classifiers correspond to systems that detect spam by subscribing to commercial blacklists or by employing content-based filters. On the other hand, normal nodes can classify connections only after their users read the email. That is, the normal classification can be delayed based on the behavior of the users (how frequently they check their email.) 10% of honest SocialFilter nodes have the ability of instant classification and the average delay of the normal classification is 2 hours [18]. The direct trust between users that are friends is initialized to a random value in $[0,\ 1]$. The number of pretrusted nodes used is 100.

4.2 Resilience to Colluders and Sybils

We consider attack scenarios under which spammers collude to evade SocialFilter and Ostra, as well as to disrupt email communication from legitimate hosts. We assume that spammers are aware of each other, which is reasonable if the spammers belong to the same botnet. In particular, to attack SocialFilter a spammer submits a report $\{[\text{spammer report}]\ \text{s},\ 0\%\}$ for each of the other spammers s in the network. Also, when a spammer receives a connection from a legitimate host l, it submits $\{[\text{spammer report}]\ \text{l},\ 100\%\}$ to induce SocialFilter to block l's emails. To attack Ostra, each spammer classifies a legitimate email and a spam email connection as unwanted and legitimate, respectively.

Figure 2(a) shows the percentage of blocked spam and legitimate email connections in SocialFilter and Ostra as a function of the percentage of nodes in the network that are colluding spammers. Regarding the effectiveness in blocking spam connections, SocialFilter outperforms Ostra, especially when the portion of colluding spammers is less than 0.5%. We also observe that Ostra achieves almost the same effectiveness in blocking spam connections regardless of the percentage of colluding spammers, whereas in SocialFilter, the percentage of blocked spam decreases with the percentage of colluders.



(a) (b)

Figure 2: (a) Percentage of blocked spam and legitimate email connections for SocialFilter (SF) and Ostra as a function of the portion of colluding spammers; (b) Percentage of blocked spam and legitimate email connections for SocialFilter (SF) and Ostra as a function of the number of Sybils created per spammer. The percentage of spammer nodes is 0.5%. Results for SocialFilter that does not employ identity uniqueness (IU) are also included.

We can also see in Figure 2(a) that Ostra suffers from a substantial false positive rate when the percentage of spammers is greater than 0.1%. When the percentage of spammers is 1% (500 spammers), around 0.8% of legitimate emails are blocked. We can attribute Ostra's high false positive rate to the following. In SocialFilter, a node blocks an email sender only if it has been explicitly reported as spammer by the repository. On the other hand, Ostra blocks links (credit balance goes out of bounds) in the socials path used by a spammer, and some honest nodes cannot send email because those links are included in all the social paths used by those honest nodes.

SocialFilter also suffers from a non-zero false positive, however it is substantially less than for Ostra. This is because colluding spammers have very low direct trust to other honest users as their reports are different from those of honest nodes. As a result, the reporter trust for spammers is lower, resulting in their reports to be mostly ignored by honest nodes.

We also consider the case in which colluding spammers create Sybils. These Sybils form a cluster that is directly connected to their creator spammer node. The purpose of the Sybils is to decrease the belief of the repository in the spammer node being malicious and to increase the belief in an honest node being spammer. In addition, Sybils allow the spammer to send messages from many different sources, enabling him to further evade defenses. At the start of the SocialFilter simulation, Sybils send positive spam reports for all other spammer nodes (including the Sybils.) Honest nodes may send legitimate email to spammer nodes but not to their Sybils. When a spammer node receives legitimate email from an honest node, the spammer reports the good user as a spammer and so do all the Sybils of the spammer. 10% of all Sybils act as spammers, sending spam messages at the same rate as their creator. In the simulation for Ostra, Sybil nodes classify a legitimate email and a spam email connection as unwanted and legitimate, respectively.

Figure 2(b) shows the percentage of blocked spam and legitimate email connections as a function of the number of Sybils per spammer in the network. In SocialFilter, Sybil users gets very low identity uniqueness, which becomes even lower as the number of Sybil users per spammer increases. As a result, we can see in Figure 2(b) that SocialFilter is resilient to this attack. In Ostra, Sybil

spammers cannot send spam because the few social links that connect the creator of the Sybils with the rest of the network become blocked. We observe that when each spammer creates more than 100 Sybils, Ostra is able to block more spam than SocialFilter. However, Ostra still suffers from a higher false positive rate.

Figure 2(b) also shows the case in which SocialFilter does not employ identity uniqueness ("SF-Spam/Legitimate-without IU"). As can be seen, attackers are very effective in manipulating the system in this case. SocialFilter without identity uniqueness cannot block a substantial percentage of spam, while it blocks a high percentage of legitimate email. This result profoundly illustrates the importance of integrating identity uniqueness in the spammer belief computation (Equation 3.)

5 Related Work

SocialFilter is inspired by prior work on reputation and trust management systems [14]. Well-known trust and reputation management systems include the rating scheme used by the eBay on-line auction site, object reputation systems for P2P file sharing networks [15, 25] and PageRank [8]. In contrary to the above systems, our system incorporates social trust to mitigate false reporting and Sybil attacks. EigenTrust [15] provides trust values that enable a system to rank users based on their trustworthiness. However, this value cannot be explicitly interpreted as the belief in a node being honest.

SocialFilter is similar to IP blacklisting services such as SpamHaus [6], DShield [2, 28], CloudMark [1] and TrustedSource [5] in that it employs a centralized repository. Currently, IP blacklisting relies on a relatively small number (in the order of a few hundreds or thousands) of reporters. Reporters submit their attack logs to the centralized repositories, and the repository synthesizes blacklists based on the attack logs. SocialFilter differs in that it automates the process of evaluating the trustworthiness of the reports. Thus it does not incur the management overhead of traditional IP blacklisting services. It can therefore scale to millions of reporters.

Prior work also includes proposals for collaborative spam filtering [3, 7, 29, 30]. Kong et al. [16] also consider untrustworthy reporters, using Eigentrust to derive their reputation. However, these solutions only enable classifying the contents of emails and not the source of spam. This requires email servers to waste resources on email reception and filtering. Similar to SocialFilter, RepuScore [22] is also a collaborative reputation management. Unlike SocialFilter, RepuScore does not employ sybil-resilient and transitive trust inference, which results in the trust values being susceptible to manipulation.

6 Conclusion

We presented SocialFilter, the first collaborative spam mitigation system that assesses the trustworthiness of spam reporters by both auditing their reports and by leveraging the social network of the reporters' administrators. SocialFilter weighs the spam reports according to the trustworthiness of their submitters to derive a value that reflects the system's belief that a host is spamming.

The design and evaluation of SocialFilter illustrates that: a) we can improve the reliability and the attack-resilience of collaborative spam mitigation by introducing Sybil-resilient OSN-based trust inference mechanisms; b) using social links to obtain the trustworthiness of spammer reports can result in comparable spam-blocking effectiveness with approaches that use social links to rate-limit spam (e.g., Ostra [18]); c) SocialFilter yields less false positives than Ostra.

Acknowledgements

This work was funded in part by an NSF CAREER Award CNS-0845858 and Award CNS-0925472. We thank the anonymous reviewers for their helpful feedback and suggestions.

References

[1] Cloudmark. `www.cloudmark.com/en/home.html`.
[2] Cooperative Network Security Community. `http://www.dshield.org/`.
[3] Distributed Checksum Clearinghouses Reputations. www.rhyolite.com/dcc/reputations.html.
[4] Facebook connect. `developers.facebook.com/connect.php`.
[5] Secure Computing, TrustedSource. `www.securecomputing.com/index.cfm?skey=1671`.
[6] The SpamHaus Project. `www.spamhaus.org/`.
[7] J. Attenberg, K. Weinberger, A. Dasgupta, A. Smola, and M. Zinkevich. Collaborative Email-Spam Filtering with the Hashing Trick. In *CEAS*, 2009.
[8] S. Brin and L. Page. The Anatomy of a Large-scale Hypertextual Web Search Engine. In *Computer Networks and ISDN Systems*, 1998.
[9] A. Cheng and E. Friedman. Sybilproof Reputation Mechanisms. In *P2PEcon*, 2005.
[10] J. R. Douceur. The Sybil Attack. In *IPTPS*, March 2002.
[11] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. A Walk in Facebook: Uniform Sampling of Users in Online Social Networks. In *INFOCOM*, 2010.
[12] E. Gray, J.-M. Seigneur, Y. Chen, and C. Jensen. Trust Propagation in Small Worlds. In *LNCS*, pages 239–254. Springer, 2003.
[13] R. K. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of Trust and Distrust. In *WWW*, 2004.
[14] K. Hoffman, D. Zage, and C. Nita-Rotaru. A Survey of Attack and Defense Techniques for Reputation Systems. In *ACM Computing Surveys*, 2008.
[15] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *WWW*, 2003.
[16] J. S. Kong, B. A. Rezaei, N. Sarshar, V. P. Roychowdhury, and P. O. Boykin. Collaborative Spam Filtering Using e-mail Networks. In *IEEE Computer*, 2006.
[17] J. Leskovec and C. Faloutsos. Sampling from Large Graphs. In *SIGKDD*, 2006.
[18] A. Mislove, A. Post, P. Druschel, and K. P. Gummadi. Ostra: Leveraging Social Networks to Thwart Unwanted Traffic. In *NSDI*, 2008.
[19] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. In *Computer Networks*, 1999.
[20] A. Ramachandran and N. Feamster. Understanding the Network-level Behavior of Spammers. In *ACM SIGCOMM*, 2006.
[21] G. Singaraju and B. B. Kang. RepuScore: Collaborative Reputation Management Framework for Email Infrastructure. In *USENIX LISA*, 2007.
[22] G. Singaraju, J. Moss, and B. B. Kang. Tracking Email Reputation for Authenticated Sender Identities. In *CEAS*, 2008.
[23] M. Sirivianos, X. Yang, and K. Kim. SocialFilter: Introducing Social Trust to Collaborative Spam Mitigation. `www.cs.duke.edu/~msirivia/publications/socialfilter-tech-report.pdf`, 2010.
[24] D. N. Tran, B. Min, J. Li, and L. Subramanian. Sybil-Resilient Online Content Rating. In *NSDI*, 2009.
[25] K. Walsh and E. G. Sirer. Experience with an Object Reputation System for Peer-to-Peer Filesharing. In *NSDI*, 2006.
[26] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipko. Spamming Botnets: Signatures and Characteristics. In *ACM SIGCOMM*, 2008.
[27] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao. A Near-Optimal Social Network Defense Against Sybil Attacks. In *IEEE S&P*, 2008.
[28] J. Zhang, P. Porras, and J. Ullrich. Highly Predictive Blacklisting. In *USENIX Security*, 2008.
[29] Z. Zhong, L. Ramaswamy, and K. Li. ALPACAS: A Large-scale Privacy-Aware Collaborative Anti-spam System. In *IEEE INFOCOM*, 2008.
[30] F. Zhou, L. Zhuang, B. Y. Zhao, L. Huang, A. D. Joseph, and J. Kubiatowicz. Approximate Object Location and Spam Filtering on Peer-to-Peer Systems. In *ACM/IFIP/USENIX Middleware*, 2003.