Trusting Spam Reporters: A Reporter-based Reputation System for Email Filtering

ELENA ZHELEVA

University of Maryland, College Park

ALEKSANDER KOŁCZ

Microsoft Live Labs

LISE GETOOR

University of Maryland, College Park

Spam is a growing problem; it interferes with valid email and burdens both email users and service providers. In this work, we propose a reactive spam-filtering system based on reporter reputation for use in conjunction with existing spam-filtering techniques. The system has a trust-maintenance component for users, based on their spam-reporting behavior. The challenge that we consider is that of maintaining a reliable system, not vulnerable to malicious users, that will provide early spam-campaign detection to reduce the costs incurred by users and systems. We report on the utility of a reputation system for spam filtering that makes use of the feedback of trustworthy users. We evaluate our proposed framework, using actual complaint feedback from a large population of users, and validate its spam-filtering performance on a collection of real email traffic over several weeks. To test the broader implication of the system, we create a model of the behavior of malicious reporters, and we simulate the system under various assumptions using a synthetic dataset.

Categories and Subject Descriptors: H.1 [**Information Systems**]: Models and Principles

General Terms: Algorithms

Additional Key Words and Phrases: spam filtering; reputation systems; trust.

Author's address: E. Zheleva, Computer Science Department, AV Williams Bldg., University of Maryland, College Park, MD, 20742.

1. INTRODUCTION

Email spam is a growing problem for the Internet community. Spam interferes with valid email, and it burdens both email users and email service providers (ESPs). It is a source of annoyance, it adversely affects productivity, and it translates to significant monetary costs for the industry (e.g., bandwidth, storage, and the cost of supporting filtering infrastructures). Also, for some categories of spam, such as *phish scams*, the costs for users may be even greater. Many solutions to tackle spam have been proposed both by the industry and by the research community. While various automated and semi-automated spam-filtering approaches exist, there are still many challenges to overcome and the goal of eradicating spam remains elusive [Fawcett 2003].

From the perspective of a large ESP, spam can often be considered as a series of dedicated campaigns, targeting large numbers of the ESP's customers with very similar messages. In that context, spam-filtering techniques can be divided into either predictive or reactive [Kołcz et al. 2004; Hall 1999]. Predictive techniques develop a general model of spam vs. non-spam, based on a large number of examples, with the assumption that the training data represents a fairly good match to the distribution of future mail. Unfortunately, this is rarely the case, and building predictive systems that cannot be outwitted by sophisticated spammers is virtually impossible. Reactive approaches to spam filtering attempt to track individual campaigns and may thus target narrow categories of spam, assuming that once such a narrow category becomes observable it is likely to persist in similar form for some period of time. Campaign-oriented filtering relies on the ability to identify volumes of similar messages in the email stream and on some way of distinguishing between legitimate emails and spam campaigns. Such data can often be identified through user complaints or reports. Unfortunately, when relying on complaint volume, by the time the system responds by acting upon such messages, the spam campaign has already affected many users.

This paper proposes a reactive spam-filtering system, based on reporter reputation to complement existing spam-filtering techniques. The challenge considered here is the reduction in response time of the reactive system; i.e., recognizing a

spam campaign at an earlier stage, thus reducing the costs that users and systems incur. Specifically, we evaluate the spam coverage (also known as recall, detection rate or sensitivity) of a reputation system that utilizes the spam-report feedback of trustworthy users. Although spam-filtering systems relying on reporters' reputations have been recently proposed [Prakash and O'Donnell 2005], few details of their operation have been made available. One of the contributions of our work is a detailed algorithm for spam filtering based on reporter reputation.

In our proposed system for spam filtering, the reports of reliable users are assigned a higher validity than those of other users. Therefore, a crucial element in the spam-filtering algorithm of the system is how to identify these trustworthy users. Ideally, a few reliable reports would be enough to flag a campaign as spam. However, such a criterion is vulnerable to malicious users who could gain trust only to abuse it later, particularly in the case of large, free email providers. The second important contribution of this work is our proposed reputation-maintenance system, and an assessment of its vulnerabilty to attack.

Thirdly, our work is the first to demonstrate the practicality of a reporter-based reputation system for spam filtering. We extensively analyze and evaluate our proposed system, using complaints received from a large population of real users. We present the results from simulating our reputation system for spam filtering over a period of time. The evaluation dataset includes spam reports collected over several months from the users of a large ESP. We evaluate how effectively our algorithm reduces campaign detection time and increases spam coverage of the existing spam filter.

The analysis on the real-world data was conducted in an experimental setting, void of any malicious users trying to game the system. Our fourth contribution is testing the broader implication of the system by creating a simple model of the behavior of malicious users and generating a synthetic dataset to match the model. We simulate the reputation system on that dataset under different possible attack scenarios to provide guidelines on how to optimize the parameters of the system. We show that unless a malicious user has a hold of the majority of the ESP accounts, the damage to the system can be kept within reasonable limits.

We recognize the key desirable properties of a reporter-based reputation system for spam filtering to be:

(1) timely and accurate recognition of a spam campaign,

(2) automatic maintenance of a reliable userset,

(3) having a set of guarantees on the system vulnerability.

Each of the desirable properties drives how the system filters spam, how it maintains user reputations, and how the systems vulnerability to a spam attack is measured. Next, we describe related work on spam filtering and trust (Section 2). Then, we define spam campaigns and a spam-filtering procedure based on user reputation in Section 3. Section 4 describes the reputation-maintenance system. Based on the spam-filtering and reputation-maintenance setups, we assess the system's vulnerability to a malicious attack in Section 5. Section 6 contains guidelines for evaluating the system, together with experimental results from real-world and synthetic data. We summarize our work and identify future research directions in Section 7.

2. BACKGROUND AND RELATED WORK

This section consists of a brief overview of spam-filtering research and summarizes related work in trust and reputation systems.

2.1 Spam Filtering

Spam-filtering techniques can be divided into three broad categories [Kołcz et al. 2006], based on: sender reputation, email-header analysis and analysis of message content. In a sender-based reputation framework, senders are classified as "spammers" or "good senders." This decision can be based on criteria such as the sender [Golbeck and Hendler 2004], the sender domain [Taylor 2006], or the IP address [DCC 2006]. In contrast, our approach focuses on the reputation of the spam reporters, rather than that of the senders. The email-header spam filtering is based on detecting forgery in the email header and distinguishing it from malformatting and other legitimate explanations such as those resulting from forwarding activity [Ludeman and Libbey 2006]. The third category, analysis of message content, has been of particular interest to the machine learning community [Cormack and

Bratko 2006; Yih et al. 2006; Dredze et al. 2007] where approaches such as Naïve Bayes [Sahami et al. 1998; Metsis et al. 2006; Hovold 2005; Meyer and Whateley 2004], support vector machines (SVMs) [Drucker et al. 1999; Kołcz and Alspector 2001; Rios and Zha 2004], or boosting [He and Thiesson 2007] are applied to the classification of email.

In applications of machine learning to spam detection, both batch-mode and online update models [Cormack and Bratko 2006] have been considered. Training a classifier in batch mode allows choosing from a wider range of algorithms and optimizing performance over a large quantity of training data. Conversely, unless the classifier is frequently retrained, the system may quickly fall pray to adversarial attacks [Dalvi et al. 2004]. Online learning approaches, on the other hand, allow for immediate incorporation of user feedback into the filtering function, but tend to be more difficult to tune, and the number of efficient algorithms is limited. Naïve Bayes tends to be most popular [Metsis et al. 2006], although online variants of logistic regression [Goodman and Yih 2006] and SVMs [Sculley and Wachman 2007] have been proposed. In either approach, changes to the classification function may require a significant number of new examples, especially if the amount of data used to derive the current models was already very large. However, sometimes large quantities of highly similar spam (i.e., a campaign) are sent within a relatively short period of time. The diversity of messages within a campaign may be too low to effectively adjust the decision function quickly enough. It is therefore convenient to consider augmenting the operation of a conventional spam filter with one that tracks high-volume spam campaigns and attempts to eliminate those mailings only.

One of the problems in automating spam classification is the lack of a consensus definition for spam [Taylor 2006; Prakash and O'Donnell 2005]. What some people consider spam may be considered solicited mail by others. Some email-service providers allow users to mark emails they consider spam and report them to their ESP. (Sometimes, users can also report opposite errors, i.e., when legitimate email is mistakenly classified as spam.) While that relies upon personalized definitions of spam, the cost of a large ESP's incorporating each individual's judgments into the filtering system may outweigh the benefits [Kołcz et al. 2006]. Nevertheless,

spam reports provided by users, as well as other forms of data acquisition, such as honeypot accounts [Symantec 2004; Prince et al. 2005], have been used to build and validate spam detection systems.

Of particular interest is the use of such data to track spam campaigns sent in volume over some periods of time, with a campaign assumed to consist of highly similar and often near-duplicate messages. In that context, when many users report nearly identical emails as spam, one can reasonably label the nature of a campaign by the volume of reports received. A key requirement to the success of such a scheme is the ability to identify emails belonging to the same campaign, despite small or irrelevant differences (some tactically inserted by the spammer to complicate detection). The problem can be otherwise described as near-duplicate message detection, which has received considerable attention in the field of Information Retrieval [Chowdhury et al. 2002; Broder 1997; Henzinger 2006]. In the email domain, near-replica (and sometimes exact-replica) based spam detection [Hall 1999; Kołcz et al. 2004] has not been studied as extensively as other content-based methodologies. Nevertheless, such techniques have been applied in practical systems (e.g., Distributed Checksum Clearinghouse [DCC 2006] and Vipul's Razor [Prakash and O'Donnell 2005]).

In essence, a duplicate-based spam detector decomposes each message into one or more fingerprints or signatures, and uses them for indexing, as well as for computing message similarity. Operationally, a few signature-based hash-table lookups are used to determine whether highly similar messages have been labeled spam and to act on an incoming message accordingly. Fingerprinting algorithms differ in the attributes they use for signature computation (e.g., direct message content, message blocks, and subsets of text features). The systems also differ in the number of signatures per message (e.g., [Prakash and O'Donnell 2005] report on the use of 6 different fingerprinting algorithms). Relying on just one signature is convenient for computational reasons; but, because of the inherent brittleness of signatures to modifications of content, systems based on several signatures are usually more robust. The advantage of applying signatures is two-fold. First, it is a concise and quickly computable representation of messages with near-duplicate content.

Second, it provides an initial clustering of the messages, which can then be processed further by more complicated clustering schemes, such as those based on cosine similarity [Kołcz et al. 2004].

Signature-based deduplication is a form of clustering. In that context it has been suggested that the stream of all incoming emails is clustered to identify high-density spikes in the content distribution, which are likely to correspond to email campaigns [Yoshida et al. 2004]. That requires an ESP to have a dedicated setup in the path of message delivery and is not easy to experiment with in a research environment. Another use of clustering is to verify cluster membership. In that context, once a cluster signature becomes known (e.g., via user reports), it is easy to determine whether an arbitrary message falls into the same cluster.

2.2 Trust and Reputation

The multi-agent system research community models trust and reputation in the context of large-scale, open, distributed systems [Ramchurn et al. 2004]. The models involve interaction between autonomous agents, using specific mechanisms and protocols, studying individual-level and system-level trust. Individual-level trust models allow an agent to evaluate the trustworthiness of another agent. In those models, trust can be formed in three ways: by direct interaction [Witkowski et al. 2001], without, or prior to, direct interaction (e.g., *Ebay* reputation system [Resnick and Zeckhauser 2002]), or by socio-cognitive means. System-level trust is based on protocols that the system creates to ensure that user actions can be trusted.

The trust model of our work falls into the category of user-level trust models, and it involves two types of agents: the spam-filtering system and the email users. It models user trustworthiness, based on direct interaction, i.e., the spam-reporting behavior of each user. It differs from the *Ebay* reputation system (and others like it) [Resnick and Zeckhauser 2002; Resnick et al. 2000] in that the *Ebay* user reputation is built by ratings assigned by other users, whereas in our system a single agent (the system itself) calculates the user reliability. Besides direct interaction, our system uses collaborative reporting patterns to update user reputation. By designating a received email as spam, trustworthy users can indirectly determine the trustworthiness of other users who may agree or disagree with the trustworthy

users.

The closest approach to the one which we take is the Cloudmark Network Classifier (CNC), an advanced commercial version of the open-source project Vipul's Razor [Prakash and O'Donnell 2005; 2007], which also uses the trustworthiness of spam reporters for spam filtering. The CNC Trust Evaluation System computes fingerprints which are invariant over small text changes in order to calculate the volume of similar email messages. Many of the details of the CNC algorithms have not been published, and it is not clear how the design characteristics, such as reputation definition and metrics, affect the system performance. Our work describes an overall reporter-based reputation spam-filtering system, and presents an in-depth comparison of trust-maintenance systems on real and synthetic data with varying parameter values.

3. SPAM CAMPAIGNS AND SPAM FILTERING

The reporter-based reputation system for spam filtering uses information from two types of incoming reports: *"this is spam"* (TIS) and *"this is not spam"* ($TINS$). The system recognizes the first type of report when a user reports as spam an e-mail initially placed by the system in a legitimate-email folder (e.g., inbox folder). It recognizes the second type of report when a user reports as non-spam an e-mail the system initially placed in the spam folder.

A *spam campaign* is a group of highly similar email messages reported by a large number of users. As mentioned in Section 2, determining the similarity is an important component to any spam-filtering system. We use an extension of the I-Match algorithm [Chowdhury et al. 2002], which is able to cluster near-duplicate emails by computing a single signature that is invariant over small changes in message content. The I-Match algorithm computes a hash representation of an email, based on its overlap with a specially constructed I-Match lexicon. The extension increases the robustness of the signature-based approaches by combining the results from K random perturbations of the lexicon [Kołcz et al. 2004]. The signature approach groups similar emails to identify spam campaigns. It is also possible to further cluster emails by grouping messages that were assigned different

signatures but have very similar content. For example, one can group emails that have a cosine similarity above some threshold. The idea is to better identify a spam campaign.

In a reporter-based reputation system for spam filtering, email signatures are computed, based on previous traffic. Determining the appropriate time window is an important design choice; picking a large window may increase coverage, but it will also increase storage costs. For now, we assume some fixed-time window, and for an email message m, we will use $\mathsf{sig}(m)$ to denote the signature of that message. The more storage resources the system has, the larger it is feasible to make the time window.

In our reporter-based reputation system for spam filtering, an incoming message m will be labeled as *"spam"* or *"non-spam"*, based on the reports of trusted users. For any set M of messages with the same signature, i.e., $\forall m_i, m_j \in M, \mathsf{sig}(m_i) = \mathsf{sig}(m_j)$, we use $\mathsf{reporters}(\mathsf{sig}(m))$ to denote the set of users who have reported any message $m \in M$ as spam. The spam score for a message $m \in M$ is computed:

$$\mathsf{score}(\mathsf{sig}(m)) = \sum_{u_i \in \mathsf{reporters}(\mathsf{sig}(m))} \mathsf{trust}(u_i) \tag{1}$$

where $\mathsf{trust}(u_i)$ is the current trust level associated with trusted user u_i (i.e., a user with a trust level higher than an established threshold). Finally, m is labeled as *"spam"* if $\mathsf{score}(sig(m))$ is above some threshold θ_{spam}. The threshold can be selected in different ways. For example, it could be a fixed number, based on the minimum number of reporters (with a trust level of 1) who can identify spam. Alternatively, the threshold could vary as a percent of the number of trustworthy users N, where trustworthy means a user with a trust level above a designated trust threshold θ_{trust}. For our experiments, we chose the percentage method.

3.1 Reputation System

Algorithm 1 shows the reporter-based reputation system. It has three main functions, FilterSpam(), which performs spam filtering, and UpgradeTrust() and DowngradeTrust() for user-trust maintenance. Those functions are explained in Subsection 3.2 and Subsection 4.3, respectively. The time window t in the trust-maintenance algorithm is to ensure that a user is not rewarded more than once in

Algorithm 1 Reporter-based Reputation System for Spam Filtering

```
1: for each time period t do
2:    Set of signatures S = ∅
3:    Set of users U = ∅
4:    for each incoming TIS report {m, u} do
5:       FilterSpam(m, u)
6:    end for
7:    for each incoming TINS report {m, u} do
8:       DowngradeTrust(t, m, u)
9:    end for
10:   UpgradeTrust(t, S, U)
11: end for
```

a given time period, so that a high trust level is not assigned in a too-short period of time. The period does not need to correspond to the time window in the spam-filtering algorithm. For simplicity, the two periods are the same in Algorithm 1.

3.2 Spam-Filtering Algorithm

Algorithm 2 shows the spam-filtering algorithm in the reporter-based reputation system. Each spam report is identified by its receipt time, email message body/text m, and reporting user id u. U is a set of users reporting during a specified time period, S is a set of signatures observed during that time period.

4. REPUTATION-MAINTENANCE SYSTEM

The coverage, accuracy, and timeliness of a reporter-based reputation-maintenance system depend on the identification and maintenance of a relatively small set of reliable users, still large enough to monitor for most incoming spam attacks. User reliability will be based on the accuracy of their reports. Also, the reports of users who respond the soonest will be used to identify campaigns at an early stage. Users who report correctly by agreeing with other trustworthy users are considered reliable in terms of correctness. Further, of those who report correctly, those who report most promptly are most helpful. Users with high email and reporting activity are especially valuable for keeping the userset small and the feedback timely. Users

Algorithm 2 Reporter-based Reputation System: Spam-Filtering Algorithm

FilterSpam(m,u)

1: **if** $(u \notin U)$ **then**
2: $U = U \cup u$
3: **end if**
4: $s = \mathsf{sig}(m)$
5: **if** $(\mathsf{trust}(u) > \theta_{trust})$ **then**
6: **if** $(s \notin S)$ **then**
7: $S = S \cup s$
8: $\mathsf{spam}(s) = false$
9: $\mathsf{score}(s) = 0$
10: $\mathsf{reporters}(s) = \emptyset$
11: **end if**
12: **if** $(\mathsf{spam}(s) == false)$ **then**
13: $\mathsf{reporters}(s) = \mathsf{reporters}(s) \cup u$
14: $\mathsf{score}(s) += \mathsf{trust}(u)$
15: **if** $(\mathsf{score}(s) > \theta_{spam})$ **then**
16: $\mathsf{spam}(s) = true$
17: **end if**
18: **end if**
19: **end if**

who do not report promptly are less valued though they may report spam correctly. Because the reporting behavior of users varies over time, the set of reliable reporters must change over time as well.

We also need to design a reputation-maintenance system that is not vulnerable to attack. The actual number of needed trustworthy users varies among ESPs and it depends on the behavior of their users. To apply the system to a different ESP, it is best to estimate such system parameters by applying the system in an offline setting to data for the user reports of that ESP. Guidelines on what numbers are important and how they can be estimated are provided in Sections 5 and 6. One of the most important guidelines for choosing the size of the trustworthy userset is

an estimation of the userset spam coverage, rather than an estimation of spam not caught by the existing filter (i.e., coverage is preferred over false negatives).

4.1 Automatic Maintenance of a Large Reliable Userset

Historical data is a good source for finding an initial set of users who have proven to be reliable in the past. Given that the definition of reliable is flexible, the standard might best be set by the ESP operators who employ the system and who know the behavior of their users and what best constitutes *reliable*. One definition of reliable users is users who have reported spam correctly in the past; or, simply, active reporters. The definition does not need to relate directly to the reporting behavior of a user; it could be related to the user's general profile, such as how long the account has been active (independent of when the user started reporting), or amount of email the user sends and/or receives. The definition could also combine those or other factors.

Using an initial set, the system automatically identifies new reliable users and corrects itself if an unreliable user is taken into the community. Change in the composition of the community is necessary to ensure that the community is active and has a broad reporting coverage over the full range of spam campaigns. The community of trustworthy users grows based on collaborative filtering: people whose reports correspond to those of identified trustworthy users are themselves considered trustworthy. This allows the system to perpetuate its set of trustworthy users. That is important because the spam reporting behavior of users changes over time and because spam targets vary by campaign. It is also important to maintain a set of active reporters large enough to ensure coverage over many spam campaigns.

The precise size of the desired reliable userset should be determined empirically, depending on characteristics of the email provider and its observed user activity. This can be done by observing the spam coverage on a validation set of labeled reports for different userset sizes. The userset size that gives the best tradeoff between spam coverage and reporter-data storage overhead should be selected. In the absence of historical data, it is acceptable to start with a small reliable userset consisting entirely of the operators of the spam-filtering system and to allow the community to grow from that nucleus.

4.2 Trust Maintenance

Trust maintenance is based on the user's agreement or disagreement with what constitutes *"spam"*. The system calculates for each reporter of spam a trust score, which is recalculated upon receipt of each subsequent report. The trust score for a user u at time t is $\mathsf{trust}(u,t)$. Trust scores range from 0 to 1, a higher value indicating higher reliability that the report will coincide with the reports of the most trustworthy users. A user trust level 0 indicates that either the user has not been recognized as trustworthy by the system or has been recognized as untrustworthy by the system. There is a trust threshold, θ_{trust}, and users with scores above it are considered *trustworthy users*.

When a piece of email is recognized as spam by the community, a subset of its first reporters are rewarded. There are different ways in which the system can pick this subset. For example, a simple approach is to pick the users who have reported the email as spam within a specified time window after its first appearance. However, this approach makes the system vulnerable to rewarding spammers because they can ensure that they are the first ones to report their own spam. A better approach would tolerate the presence of malicious users; for example, the system can pick a random subset of the first reporters. [Sarmenta 2001] provides an analysis of such techniques applied to volunteer computing. In our offline experiments on real data, we used the simpler approach and rewarded the first user of each spam message.

The reward is an internal way of reflecting the positive behavior of a user (i.e., the user never sees it). As shown in Algorithm 1, a user can be rewarded only once within a given time period, which does not need to correspond to the time window considered in the spam-filtering algorithm. That restriction ensures that trust is not gained too soon. The extent to which a positive experience enhances the trust score of a user, α, is a number between 0 and 1. The number is large when the system builds trust in its users easily, and small otherwise. The trust upgrade function is

$$\mathsf{trust}(u,t) = \mathsf{trust}(u,t-1) + \alpha \cdot (1 - \mathsf{trust}(u,t-1)). \tag{2}$$

On the other hand, when a user reports an email as non-spam but the community has identified it as spam, trust in the user diminishes. A user's trust is downgraded

as many times as (s)he submits such a report. That is to ensure that the system can make a timely response to a user who begins behaving suspiciously. The degree to which a negative experience (i.e., an incorrect report) lowers the trust score of a user, β, is also a number between 0 and 1. The number is large when the system can lose trust in its users easily, and small otherwise. The trust downgrade function is

$$\mathsf{trust}(u,t) = \mathsf{trust}(u,t-1) - \beta \cdot \mathsf{trust}(u,t-1). \tag{3}$$

We designed the trust-upgrade and trust-downgrade functions to correspond to the intuition that trust is hard to gain and easy to lose when α is low and β is high. Witkowski et al. used those functions for creating a trust model in a game-theoretic framework. In their model, buying and selling agents interacted directly, and every agent computed and maintained trust levels of the other agents in order to choose with whom to trade [Witkowski et al. 2001].

Trust is increased when a user's judgment is consistent with that of the community; and decreased when it is not, and recent experiences are given greater weight. Higher values of α lead to susceptibility to only few positive experiences, and higher values of β lead to harsher penalizing of disagreeing users. α and β could be optimized by picking a validation set of report data, and running the system with different values of α and β to see which values allow the reliable userset to grow to a desired size and keep a steady spam coverage over time. Ideally, α and β would be set to the values resulting from the validation set, but in practice, they may need to be changed over time to reflect the user-pool dynamics.

Alternative trust functions. Giving more weight to recent experiences is rooted in the fact that the upgrade function does not consider previous steps. Only the last trust value is taken into account, and the previous trust values are considered only implicitly. In the trust upgrade function, there is no reference to what has happened in previous steps or whether the user has been continuously upgraded or downgraded until time t. Therefore, the function gives weight only to the last, most recent experience. If we would like to give equal weight to k last

experiences, the trust function would be:

$$\mathsf{trust}(u,t) = \mathsf{trust}(u,t-1) + \alpha \cdot \left(1 - \frac{\sum_{i=1}^{k} \mathsf{trust}(u,t-i)}{k}\right). \quad (4)$$

The downgrade function would be similar. Depending on the desired characteristics of the individual's trust evolution over time, alternative trust functions could be considered, such as upgrading and downgrading by a constant, or by functions that give different weights to a certain number of recent experiences. [Jonker and Treur 1999] contains a further discussion on defining trust update and evolution.

4.3 Trust-Maintenance Algorithms

Algorithm 3 Reporter-based Reputation System: Trust-Upgrade Algorithm

UpgradeTrust(t,S,U)

1: **for** each $u \in U$ **do**
2: $\quad \mathsf{trust}(u,t) = \mathsf{trust}(u,t-1)$
3: $\quad$ **for** each $s \in S$ where $spam(s) == true$ **do**
4: $\quad\quad$ **if** u is one of the first reporters of s **then**
5: $\quad\quad\quad \mathsf{trust}(u,t) = \mathsf{trust}(u,t-1) + \alpha \cdot (1 - \mathsf{trust}(u,t-1))$
6: $\quad\quad$ **end if**
7: $\quad$ **end for**
8: **end for**

Algorithm 4 Reporter-based Reputation System: Trust-Downgrade Algorithm

DowngradeTrust(t,m,u)

1: **if** $(spam(\mathsf{sig}(m)) == true)$ **then**
2: $\quad \mathsf{trust}(u,t) = \mathsf{trust}(u,t-1) - \beta \cdot \mathsf{trust}(u,t-1)$
3: **end if**

Algorithm 3 and Algorithm 4 show the trust-maintenance algorithms $UpgradeTrust(t,S,U)$ and $DowngradeTrust(t,m,u)$, respectively. Note that a user's trust can be upgraded at most once during a time period, but it will be downgraded for each inaccurate *TINS* report they have made.

If the collective intelligence of the community is occasionally wrong, the trust levels of disagreeing users fall and they are temporarily excluded from the trustworthy userset. However, if they continue to report reliably and agree with the community for the rest of the time, they will be added back to the trustworthy userset. Incidentally, users never see the effects of such downgrading, since it is part of an internal accounting system.

5. ASSESSING SYSTEM VULNERABILITY

We would like to estimate how vulnerable the system is to having malicious reporter gain a trusted reputation and wreak havoc on the system. We use the term *contamination* to refer to the presence of malicious users in the trustworthy userset. One important question is: what is the least amount of time it would take a new user to join the trustworthy userset. Using the trust upgrade function, Eq. (2), the trust value that a user would get if (s)he has been upgraded every day over the span of n days (and has not been downgraded) is:

$$\mathsf{trust}(u, t+n) = \sum_{j=0}^{n-1} \alpha \cdot (1-\alpha)^j + (1-\alpha)^n \cdot \mathsf{trust}(u, t) \tag{5}$$

If the user is new to the system, the initial trust score trust(u,0) is 0. If the trustworthy set has a minimum trust value of θ_{trust}, then the minimum number of days n needed to join the trustworthy userset is:

$$n \geq \log\left(1-\alpha\right)^{1-\theta_{trust}} \tag{6}$$

That is the least amount of time that a malicious user would need to affect reputation-system decisions.

Another important question is: what is the least number of user accounts a malicious user needs to affect the decision of the system on the "spamminess" of a certain signature. If a malicious user wants to affect the system as soon as the user's accounts join the trustworthy userset, then the user needs to report a message from m accounts such that its spam score rises above the spam threshold θ_{spam}: $\sum_{i=1}^{m} \mathsf{trust}(i, t) \geq m * \theta_{trust} \geq \theta_{spam}$. Therefore, the person needs at least

$$m \geq \frac{\theta_{spam}}{\theta_{trust}} \tag{7}$$

accounts with trust value θ_{trust} to affect the system. If the person waits until his/her accounts grow stronger – in other words have trust values larger than θ_{trust} for all the accounts – then the number of accounts needed is less. However, a malicious user has no way of directly checking the parameters of the system.

If a message is incorrectly marked as spam by the system, the damage will be two-fold. The incorrect classification will lead to classifying similar messages as spam in the future, which will lead to more false positives (more legitimate messages will be placed in the spam folder). This is not as much of an issue for personal mail as it is for bulk and mailing-list messages to which people have voluntarily subscribed. It is not an issue for personal legitimate mail because malicious users would need to report the same (or a very similar) message as spam for it to be later incorrectly marked as spam, and this is very unlikely. It could be an issue with bulk-mail/mailing-list messages because this is contested mail, meaning that some people consider it spam and others do not. If users whitelist messages from mailing lists to which they have subscribed, then they will still receive them in their inboxes rather than in their spam folders.

The second type of damage occurs when the system is misled to penalize users who report the same or similar messages as *"non-spam"* in the future. The second damage can be lessened if there is a time window in which users can be penalized after the time of spam identification.

An intentionally malicious user (a spammer) is mostly interested in reporting spam as non-spam. A system for recognizing true *TINS* reports is also desirable; however, it is harder to build because such reports by regular users are rare. It is possible that the trustworthiness of users, which was built based on true *TIS* reports, can be used to recognize true *TINS* events. Thus, the vulnerabilities of the system will hold for either event. Therefore, spammers would not be able to affect the decision over a piece of email being non-spam unless they have reported spam reliably over a certain period of time (n days).

One possible attack on the system would be a malicious user reporting every message as spam. The user would need control of a large number of email accounts that receive legitimate mail in order to generate a sufficient volume of spam complaints

to become noticeable. The user would also have to ensure that other users receive the same messages as the malicious user, which would be negatively affected if their legitimate mail were marked as spam. If the malicious user were a spammer, his main goal would be to deliver his own spam campaigns. Unless he starts reporting *TINS* on messages he wants to pass through the spam filters, his trust score may never fall but that strategy would not help him deliver his own spam campaigns. His only impact on the system would be to annoy system operators. Therefore, he could not use the same accounts to report every message as spam and to try to get his spam campaigns through the filters. Once he started disagreeing with the reliable users with his *TINS* reports, his trust score would go down. Another safeguard of the system would be automatic downgrade of trust for signatures, which the system operators might think are legitimate.

The vulnerability assessment allows a system designer to set the system parameters to reflect an acceptable level of attack exposure. The system parameters that need to be set are θ_{spam}, θ_{trust}, α and β. The thresholds θ_{spam} and θ_{trust} can be computed from Eq. (6) and Eq. (7) once α and β are set and the system designer decides the allowed vulnerability of the system by setting the values of n and m. If the designer chooses to be very conservative with n, the growth of the reliable userset will be very conservative too, and it would take a long time for a new user to join the system. On the other hand, a high value for n means that a malicious user (usually, a spammer) would need to be very patient and act reliably over the span of n days in order to join the trustworthy userset and become able to affect the system. The value of m can be determined if there is a restraint on how many accounts a user can create from a particular IP. The size of the reliable userset varies with time but the designer can set its initial value by the procedure explained in Subsection 4.1.

6. EXPERIMENTAL SETTING

We did an extensive experimental evaluation of the effectiveness of the reputation system under two scenarios. For the first one, we used a real-world dataset with spam reports received by a large ESP. The questions we were interested in were

concerned with the reporting behavior of the users, the spam coverage of the reputation system, and the response time of the system. The experiments were done in an offline setting, not in a production setting. To test the system in the presence of malicious users, we also created a model of the behavior of malicious users and used it to generate a second, synthetic dataset. We experimented with the data and obtained results on how the system parameters needed to be optimized to account for malicious users and their effects on the system. Section 6.1 presents an idealized version of the evaluation strategy for a spam-filtering system, together with a discussion on what can limit its applicability. Section 6.2 describes the real-world dataset, the implementation of the reputation system for the experiments, and the measured outcome. For evaluating the impact of malicious users, Section 6.3 describes the synthetic data generation, the simulations, and their results.

6.1 Evaluation Strategy

One important measure of any spam-filtering system is its accuracy. There are four possible outcomes that the system needs to count:

. *True Positives (TP)*: a true spam message reported as spam by a reliable user.

. *False Negatives (FN)*: a true spam message, not reported as spam by a reliable user but reported as spam by other user(s).

. *True Negatives (TN)*: a true non-spam message, not reported as spam by a reliable user but reported as spam by other user(s).

. *False Positives (FP)*: a true non-spam message, reported as spam by a reliable user.

Note that all outcomes are counted in reported signatures not in reports (there is more than one report per signature). For example, if two reliable users reported the same message as spam correctly, the true positives increase only by one and not by two.

It is possible to assess accuracy in terms of sensitivity and specificity. *Sensitivity* (also known as recall) is the probability that a spam message will be reported as spam by the reliable userset. It is defined as the fraction of true spam messages that were reported by at least one reliable user over the true spam messages reported

by any user:

$$Sensitivity = \frac{TP}{TP + FN}. \tag{8}$$

Sensitivity is the same as the spam coverage, $Coverage_{spam}$. A sensitivity of 100% is desirable, and it means that the system classified all spam as spam. $Coverage_{spam}$ shows the reduction in error of the existing spam filtering system.

$Specificity$ is the probability that a non-spam message will not be reported as spam by the reliable userset. It can be described in terms of the coverage of the reliable userset over non-spam:

$$Specificity = \frac{TN}{TN + FP}. \tag{9}$$

The non-spam coverage is $1 - Specificity$, which is

$$Coverage_{non-spam} = \frac{FP}{TN + FP}. \tag{10}$$

High specificity and low non-spam coverage are desirable.

This is an idealized evaluation setting, since it requires a dataset with user reports in which all messages are labeled as either *"spam"* or *"non-spam."* Unfortunately, there is no such dataset available publicly. In the real-world dataset used in the experiments, there is a subset of the data labeled *"spam"* that allows us to approximate the sensitivity of the reputation system spam prediction. However, it is not possible to evaluate the specificity using the dataset because it contains many unlabeled messages, and it does not contain the ground truth for non-spam.

In general, manual labeling of the data is difficult because there are thousands of messages reported per day. Another factor is the lack of a universal definition of what constitutes spam. For example, bulk mail, which complies with the CAN-SPAM Act of 2003 [FTC 2003] and allows a user to unsubscribe from receiving further messages, is considered spam by many users.

6.2 Experiments with Real-World Data

We used real-world data to test the spam coverage and the response time of the reputation system and to get a general idea of the reporting activity of the email users. The experimental implementation consisted of an offline simulation of the reputation system. The measurements were analyzed with respect to the contribu-

tion of the reporter-based reputation system to the improved performance of the ESP's existing spam-filtering system. In other words, the effectiveness of the reputation system was measured in terms of reducing the spam-coverage error of the existing filter. First, we describe the data and the reputation system implemented for the experiments, then we present the results from the experiments. The first two experiments measured the reporting activity of a sample of users, the next three showed the spam coverage over a range of parameter settings, and the last one assessed the response time of the system.

6.2.1 *Data Description.* Our real-world data collection consists of voluntary spam reports to a large ESP for a period of 30 days. The spam reports are for messages that have not been caught by the existing spam filtering system of the ESP; therefore, the analysis is for the effectiveness of a reputation system on catching spam that reached users' mailboxes. Some of the reported messages in the dataset are marked as spam, and they provide the ground truth for true positives and false negatives.

For the sample of data made available to us, the average spam report volume is about 1.5 million per day for *TIS* reports, and 30,000 per day for *TINS* reports. In the 1.5 million spam reports, there are about 6,800 different signatures. Approximately 15,000 reports (800 signatures) per day are clearly true spam signatures, and we have used them as the ground truth for our evaluation. The true spam signatures were reported by an average of 10,000 users per day. We also use historical data for all users who have submitted true or false reports on spam and non-spam before the 30-day period.

6.2.2 *Implementation.* While the data was actual spam-report data, the evaluation was performed in an experimental setting, not in a real production setting. The initial sample of trustworthy users was selected from the users who reported a message from the largest spam campaign for a particular day, then finding those who met certain reliability criteria, based on the historical data. The criteria were that 1) the users had not reported non-spam emails as spam, 2) they had not reported spam as non-spam, and 3) they had reported spam within a certain period

after its receipt. With the initial set of users, we simulated the system over a period of 20 days, and looked at its dynamics and predictive power. We refer to the reliable userset resulting from the simulation as the *userset grown by the reputation system*.

Except in the experiments when one of the system constants was allowed to vary, the constants were $\alpha = 0.3$, $\beta = 0.5$, $\theta_{trust} = 0.3$, and $\theta_{spam} = 0.2\% \cdot N$ where N was the number of trustworthy users. Reward was given to only one user per new spam message. The spam threshold θ_{spam} was chosen experimentally so as to minimize the probability of classifying a legitimate signature as spam (less than 1% probability). In other words, we were very conservative in the classification. We considered different values for α and β, selecting those that reflected the notion that trust is easier to lose than to gain. In the presence of malicious users, the choice of α and β affects the results in a differently. We account for that in the experiments with the synthetic dataset, which assumed the presence of malicious users. We provide concrete guidelines for picking an appropriate set of values in Section 6.3.2.

We also performed experiments using additional clustering of the signatures but did not observe significant improvement in the results. While it allowed the initial trustworthy userset to grow more quickly, it showed no significant improvement over the main method at high thresholds of spam confidence and trust. The clustering was based on cosine similarity, with a threshold picked by a domain expert so that clusters mimicked spam campaigns well. The additional clustering added extra cost to the algorithm which did not lead to a better overall performance. Improvements to the campaign detection algorithm need to be made such that high level of specificity is maintained. Other clustering techniques might show better results.

6.2.3 *Experiment 1 – User Activity.* Studying the user report patterns suggested that if the reliable userset was static and did not grow, then daily reporting was low and decreased over time. Figure 1 shows the frequency of reporting, i.e., how many users from the initial reliable set report once a month, twice a month, etc. It showed that few users reported every day or nearly every day. Most users

were not very active. This result was for a sample of 2,715 users who were found to be reliable, based on historical data, and their reporting activity was for a period of 31 days. For this experiment, the reliability of users was not based on how soon they reported a spam message.

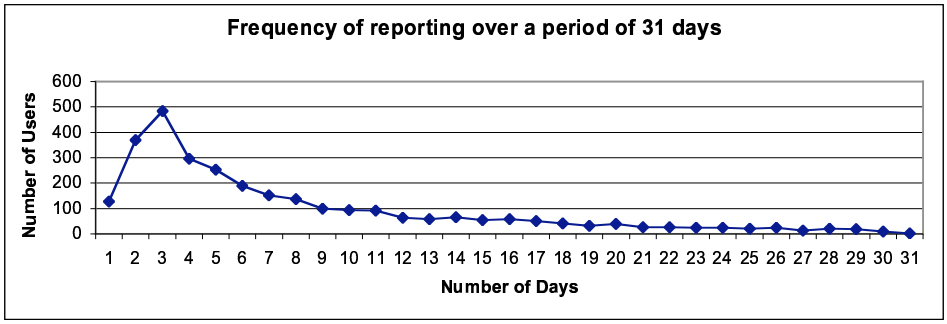

Fig. 1. For a sample set of reliable users, the graph shows the user activity (defined as the number of days they reported) over a one-month period.

Figure 2 shows how many spam reports were submitted per day by the initial set. As a percent of overall reports, the number of reports submitted by that userset tended to decline. The experiment showed that it is important to grow the userset to keep its reporting coverage high.

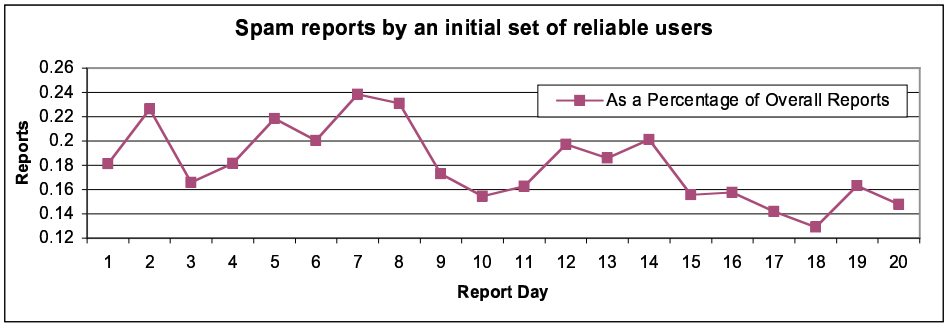

Fig. 2. Spam reports submitted by users from an initial trustworthy userset, defined for day 1.

6.2.4 *Experiment 2 – Automatic Growth.* When the reputation system was in place, an initial trustworthy set of 2,089 users grew to 2,558 users for a period of 20 days. It had a rapid growth for the first half of the period and a slow growth afterwards. Interestingly, few users were penalized over the span of the experiment. We attribute that to the fact that the system was simulated offline, and there were no malicious users trying to actively game it. In the experiment, the initial reliable users were also sampled from the spam reporters, but the sample was different from the one used in Experiment 1.

6.2.5 *Experiment 3 – Reliability and Coverage.* Next, we studied whether we could rely on the trustworthy userset by looking at its spam coverage. True spam was labeled by a domain expert from the ESP. The experiment took samples of initial trustworthy users from different days and developed them over a period of time using the spam-filtering and trust-maintenance algorithms. The period of time started on the day on which the sample was taken and ended on the 20th day of the simulation period. The experiment measured the probability that, on any particular day, a spam signature would be reported by at least one trustworthy user. We refer to that probability as *coverage*. The coverage was on the messages that had leaked through the ESP spam filter, i.e., it reflected the reduction in error of the existing filter.

The means were obtained for report-validation samples taken from days 21 to 30. Figure 3 shows that, for all the userset samples, the growing size of the userset resulted in better spam coverage. The average number of reported signatures (by any user) per day during the validation period was 473, with a standard deviation of 128.

The difference in coverage between the usersets could be attributed to the difference in size of the usersets. Figure 4 shows the grown userset sizes. However, the probabilities are rather low, and that could be explained by the fact that very few users are active on many days (see Figure 3). An interesting question is whether the combined coverage of all the grown usersets would give much better report coverage on spam. A comparison of the results displayed on Figure 5 at a confidence threshold level of 0 and Figure 3 shows that this is the case. The signature predic-

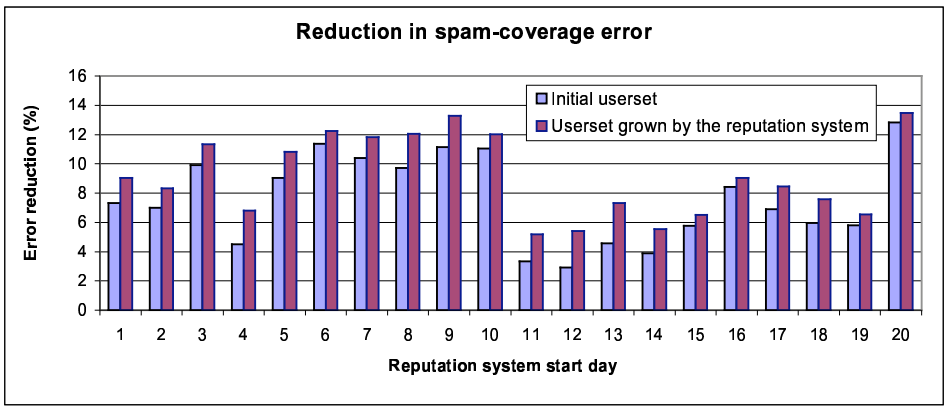

Fig. 3. Mean daily spam coverage of an initial reliable userset sample taken on the day shown on the X-axis and of the reliable userset resulting from simulating the reputation system from the day shown on the X-axis until day 20. The coverage was on the messages that have leaked through the ESP spam filter during the ten days following the simulation period, i.e., it reflects the reduction in error. This figure shows that relatively small sets of reliable users (see Figure 4) were able to reduce the error of the existing spam filter by 13% (on average). For better coverage, a larger pool of trustworthy users would be required.

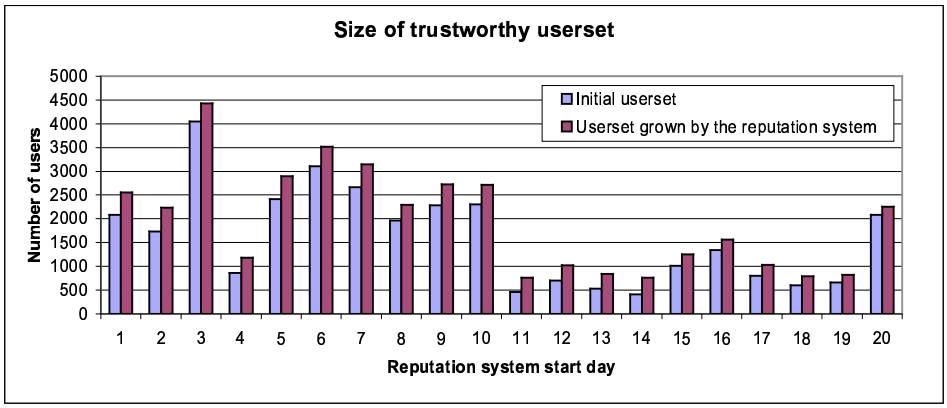

Fig. 4. Size of the usersets. Each pair of bars shows the sizes of an initial reliable userset sample picked on the day of the X-axis, and its counterpart userset resulting from developing that initial userset with the reputation system from day x until day 20.

tion probability changes from 5-14% (Figure 3) to an average of 43.8% (Figure 5) for the signature-based method. Therefore, the coverage (and trustworthy userset daily activity) can be increased by taking initial sets from different days.

Growing the usersets also affects the coverage of the signatures. The combined coverage of the 20 initial usersets is 40.8%, and it increases to 43.8% when combining the coverage of the usersets grown by the reputation system.

6.2.6 *Experiment 4 – Coverage as a Function of the Spam-Confidence Threshold.* The probability that a spam signature will be reported on a future day is equivalent to the coverage on a future day. Figure 5(a) shows the coverage of the grown userset as a function of the spam-confidence threshold. When the threshold was 0, just one report from a reliable user would cause the system to mark the message as spam. In such a case, the users covered 43.8% of the spam signatures. When the system relied on more than one trustworthy user to recognize a spam campaign, which is a realistic requirement, the coverage of the userset dropped significantly. That reiterates the fact that the spam reports were sparse, and that a large community of trustworthy users was necessary for the system to be useful in a real setting. The size of the userset was 34,803. For our dataset, that was less than 1% of the ESP accounts. However, some of the email accounts were inactive, and 34,803 represents a larger percentage of the users who opened their email every day.

6.2.7 *Experiment 5 – Coverage as a Function of Average Trust in Reporting Users.* In this experiment, we studied how the average user-trust values used in computing the spam confidence affected the userset coverage. That was similar to studying the threshold for a minimum-trust level. The initial userset coverage is not shown because all its users had a trust level of 1; therefore, its coverage did not depend on the average trust-value threshold. Figure 5(b) shows that higher average trust led to a lower coverage. That was intuitive because higher average trust meant that the userset that got to vote on spam was smaller. Even if the system were very conservative in its choice what users to trust (high average trust level), the trustworthy userset provided a reasonable spam coverage.

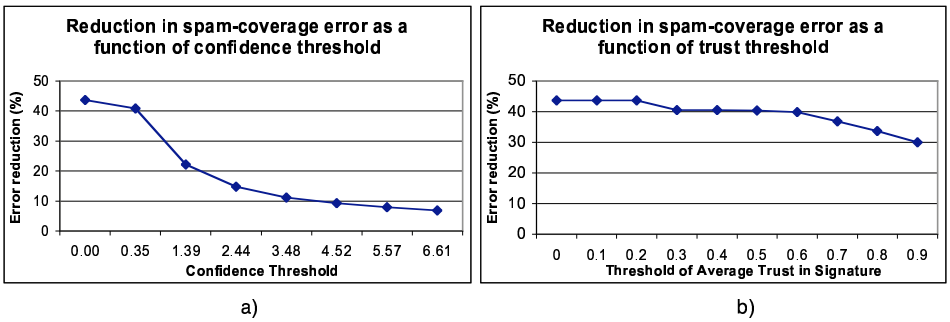

Fig. 5. Mean daily coverage by combined grown userset over the signatures for the period of 20 days with varying confidence and trust thresholds. The coverage is on the messages that have leaked through the ESP spam filter, i.e., it reflects the reduction in error. It is averaged over ten days. This figure shows that a relatively small set of 34,803 trustworthy users is able to reduce the error of the existing spam filter by 43.8%.

6.2.8 *Experiment 6 – Response Time of the System.* The objective of the reputation system is to recognize a large volume spam campaign before it has affected many users. To assess the response time of the system, we performed experiments to compare the response time of the users in the combined grown trustworthy set with the response time of the other users. In this experiment, response time was the difference in days between the receipt time of the spam email message and the time when a user reported a message as spam. For example, if a user reported a message within 24 hours, the response time was 0 days. Although we used days as the time unit, ideally we would like the system to respond within hours or minutes.

The null hypothesis was that the report samples came from the same population of users. To study that, we applied a chi-square test on the two distributions and looked up the significance of the resulting value. The two response-time distributions were taken from the reports for the same validating dates. For each day, the observations were the user-response times for the spam emails that were reported by at least one trustworthy user. The observations were separated into two samples, depending on whether or not the reporting user is from the grown trustworthy set. The chi-square test over the two sample distributions showed a highly significant difference between them. That was confirmed for both a test with *degree of freedom* = 1 (reports with 0 days response time and reports with more than or equal to 1

day response time), and for *degree of freedom* = 10 (0, 1, ..., more than 9 days) with a probability of that result occurring by chance smaller than 0.01%. Many (92%) of the trustworthy-user reports were sent within one day of the spam receipt whereas the percentage was smaller (83.5%) for the rest of the user reports.

On each day, the reports for active spam campaigns had a spam receipt date equal to that day. To see whether we could effectively use the earlier response time for active spam-campaign identification, we checked the mean response time for each of the signatures from active campaigns. We studied the distribution of the difference between the mean of the trustworthy-user response time and the mean of the other-user response time, where each observation was based on one signature received by both groups on a certain day. The trustworthy users were from the combined set, which was grown with the signature-based method. Figure 6 shows the difference in the average response time between the trustworthy set of users and the rest. For example, the number of signatures that were reported by users in the trustworthy set 1 to 2 days earlier than the rest is shown in the interval [1,2]. The distribution shows that for most signatures, the average time of response for the trustworthy userset was better. In more than 1/3 of the cases (35.6%) the trustworthy users reported at least one day earlier than the rest, whereas in only 6% of the cases, they were slower.

6.3 Experiments with Synthetic Data

In the reputation-system simulations on the real-world dataset, the system rarely downgraded users. We attributed that to the fact that the simulation was in an experimental setting, where no spammers were trying to game it. However, if the system were placed in the field, we expect that there would be gaming attempts, and the upgrade and downgrade factors α and β would play larger roles on the number of malicious users that could join the reliable userset. We created a synthetic data simulation to gain a better understanding of those factors. The simulation used a simple model that we created to mimic malicious-user behavior. The model considers malicious users who provide bogus feedback but are also trying to attain reputation. In reality, when the system is tested in the field, malicious users might use other attacks on the system, such as fragmenting the signatures, or come up

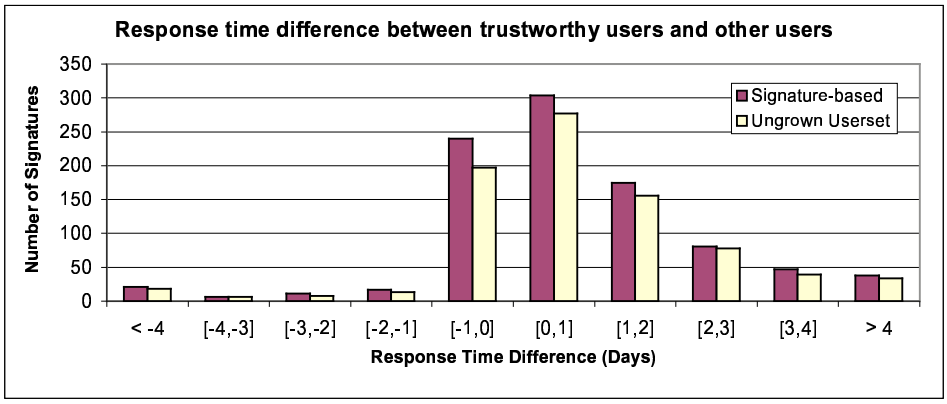

Fig. 6. The difference in mean response time for each spam signature received and reported by both types of users. The data is for spam messages that were received during days 21 to 30, and reported between days 21 and 50.

with other, yet unknown strategies. First, we describe the synthetic data generation and simulation; then, we discuss the results of varying the parameters on the contamination of the trustworthy userset and the spam coverage.

6.3.1 *Data Generator.* The data generator takes as input parameters the number of users, the percentage of: malicious users, users reporting per day, malicious users among the users reporting per day; the likelihood of: a non-malicious user submitting a true spam report, a non-malicious user submitting a false *TINS* report, a malicious user submitting a true spam report, and a malicious user submitting a false *TINS* report; α and β. The first experiment simulated the reputation system for a number of days. On each day, users were picked randomly from the respective group of users and they reported on a signature, according to the likelihood constraints. At the end of the day, the simulation ran the trust-maintenance algorithm. It upgraded all the users who submitted a true spam report. It also downgraded users as many times as they submitted a false *TINS* report. The output of the simulation was the percent of malicious users in the trustworthy userset for different parameters. The second experiment used the same data generator, and it simulated the spam-filtering algorithm for one time period. It studied how the percentage of malicious users in the trustworthy userset affected the spam coverage.

6.3.2 *Experiment 1 – The Influence of α and β on the Contamination of the Trustworthy Userset.* We studied the influence of the trust-upgrade and -downgrade factors α and β on the number of malicious users that would eventually join the trustworthy userset. α and β are parameters used in the trust-maintenance algorithm to reflect the degree to which a positive or negative experience enhances a user's trust score. Over time, the parameters directly affect the evolution of the trustworthy userset. The experiment did not intend to set the best, universally valid values of α and β, but to provide some guidelines on how to optimize their values. For the data generator, we used the following parameters: users, 1000; percent malicious users, 15%; percent users reporting per day, 10%; percent malicious users of users reporting per day, 50%; likelihood of a non-malicious user submitting a true spam report, 80%; likelihood of a non-malicious user submitting a false *TINS* report, 20%; likelihood of a malicious user submitting a true spam report, 30%; likelihood of a malicious user submitting a false TINS report, 70%; $\alpha = 0.1$ and $\beta = 0.9$. The parameter values were chosen to match the intuition that malicious users reported more actively than non-malicious ones, and that even though they were more interested in submitting false *TINS* reports, they also had interest in building good reputation. Those parameters were used in all the experiments except those where α, β, likelihood of a malicious user submitting a true spam report, and the percent of malicious users were allowed to vary. Running the system for 1,000 time periods (e.g., days) showed that the size of the reliable userset increased until a certain time point (usually around time period 100), after which it leveled off and fluctuated very little. Increasing the number of users, and keeping the other parameters the same, showed the same pattern, except that the number of reliable users stabilized at a later time.

Varying α and β showed that the best way to keep the malicious users out of the reliable userset was to keep α very low and β very high. Table I shows that at different α and β, the trustworthy userset evolved to contain different percents of malicious users. Table I b) shows the percentage of the non-malicious users represented in the trustworthy userset. The initial trustworthy userset contained 2.4% of the non-malicious users. The trust threshold θ_{trust} is 0.9. The results are

$\alpha \downarrow$ \ $\beta \rightarrow$	0.1	0.5	0.9
0.1	2	0	0
0.3	13	5	4
0.5	14	10	8

a)

$\alpha \downarrow$ \ $\beta \rightarrow$	0.1	0.5	0.9
0.1	99	83	78
0.3	100	95	93
0.5	100	97	96

b)

Table I. Table a) shows that, depending on α and β, the trustworthy userset evolved to contain different percents of malicious users. Table b) shows the percentage of the non-malicious users in the trustworthy userset. The initial trustworthy userset contained 2.4% of the good users in the final trustworthy userset. The trust threshold θ_{trust} was 0.9. The results showed that a low level of α and a high level of β were desirable.

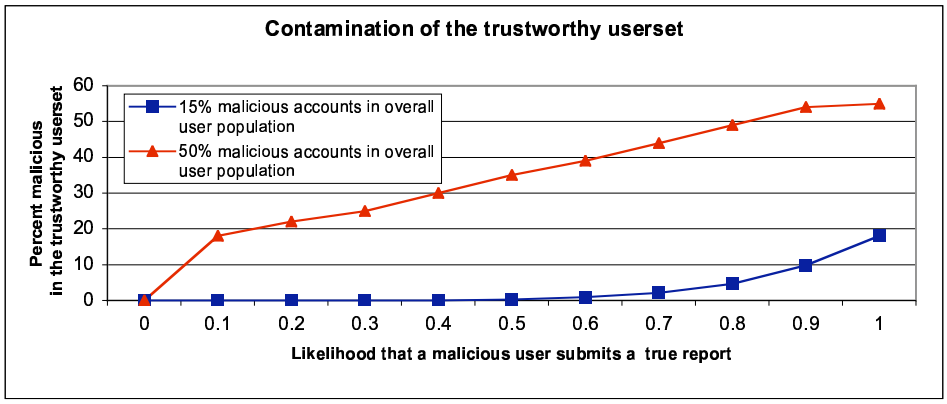

Fig. 7. This graph presents the relationship between the likelihood of a malicious user to submit a correct report and the percent of malicious users in the trustworthy userset. If the malicious users are very likely to submit correct reports, then the trustworthy userset can get contaminated even at low $\alpha = 0.1$ and high $\beta = 0.9$. However, even under this extreme assumption, the percent of malicious users in the trustworthy userset is still limited (less than 20%) when the percent malicious accounts in the overall user population is low.

averaged over 10 trials.

Varying the percentage of correct reports submitted by malicious users showed that keeping α low and β high could not prevent malicious users out of the reliable userset. Figure 7 shows that damage to the system can be kept within reasonable limits unless the malicious users comprise the majority of the ESP accounts. When the likelihood that a malicious user reports correctly rises, so does the percentage

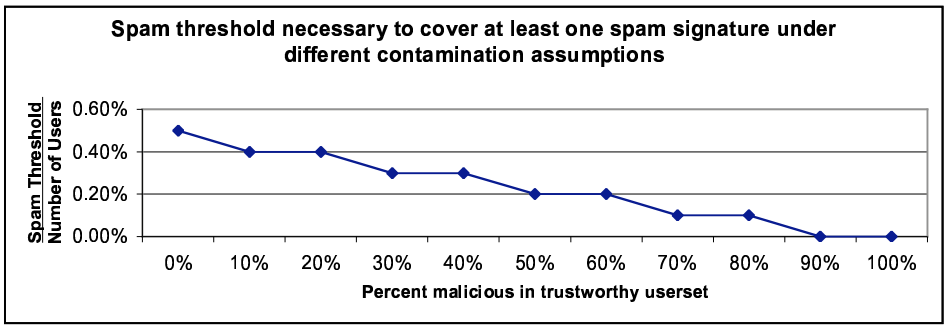

Fig. 8. The graph presents the relationship between the percentage of malicious users in the trustworthy userset and the minimum spam threshold (as a percent of the trustworthy userset size) necessary for the reputation system to be effective. When the trustworthy userset is contaminated with a lot of malicious users who stop reporting spam, then the reputation system would not have any spam coverage at high spam threshold levels. Therefore, a fixed spam threshold and an influx of malicious users in the trustworthy userset would make the system ineffective.

of malicious users in the reliable userset. It also shows that, even if we do not know the exact behavior of malicious users, their percentage in the trustworthy userset is low. In addition to the more realistic case, in which 15% of the ESP user accounts were owned by malicious users, we also studied the extreme case, in which 50% of the accounts were owned by malicious users. This was to distinguish between ESPs in which malicious users can easily create many accounts automatically, and those in which there are different authentication procedures that make it difficult (the account is paid, it is connected to a cell phone, etc.).

The result of this experiment illustrates the intuition that when malicious users consistently report correctly, agreeing with the trustworthy users (100% correct *TIS* reports by the malicious users), they can eventually infiltrate the reliable userset. However, that does not benefit malicious users until they report spam as non-spam. Then, if "many" other reliable users report the same piece as spam, the malicious users will eventually get downgraded. The next experiment explores how many is "many" at different contamination levels, where contamination refers to a large percentage of malicious users in the trustworthy userset.

6.3.3 *Experiment 2 – The Influence of the Malicious Users on Spam Coverage.* The primary objective of malicious users joining the trustworthy userset is to use their high trust to make the system accept *TINS* reports as true when they are not. However, that is not a part of the reputation system, so they cannot affect it. The main damage that malicious users can inflict, once they contaminate the trustworthy userset, is to render the reputation system ineffective for spam filtering. However, they would not damage the capabilities of the existing spam filter with which the reputation system would operate. If the trustworthy userset consists mostly of malicious users who have submitted correct spam reports over time, at some point they could stop reporting spam. That would reduce the spam coverage. If the spam threshold θ_{spam} is not adjusted accordingly, the few non-malicious users in the trustworthy userset would not be able to enhance the existing spam-filtering system.

This experiment assesses how the percentage of malicious users in the trustworthy userset affects the reputation system. The data generator used was the same as in the previous experiment. In addition, we simulated the spam-filtering algorithm for one time period, instead of simulating the trust-maintenance algorithm over a few time periods. We considered 1,000 trustworthy users and 10 true spam signatures reported per day. The reporting activity per time period was the same as in the previous experiment. We studied how the spam threshold θ_{spam} needed to be adjusted with the variation of the percentage of malicious users in the trustworthy userset. Figure 8 shows the relationship between the percentage of malicious users in the trustworthy userset and the spam threshold. High contamination of the trustworthy userset leads to low coverage at high spam-threshold levels. Therefore, a fixed spam threshold and an influx of malicious users renders the system ineffective.

7. CONCLUSIONS AND FUTURE WORK

We described a framework for a reporter-based reputation system for spam filtering. The system includes a trust-maintenance component, in which users gain and lose reputation, depending on their spam-reporting patterns. The filtering component

uses the reports of highly reputable reporters for spam removal. We describe several desiderata for a reputation-based spam-flitering system, and a set of guarantees on the system's vulnerability. We evaluated our proposed framework, using actual complaint-feedback data from a large population of real users, and for a period of several weeks, validated its spam-filtering performance on a collection of real email traffic. The evaluation suggested that a reporter-based reputation system can be used for early prediction of spam – something that recognition of spam by volume usually lacks. It also showed that because of the low frequency of user reporting, the system needs to have a relatively sophisticated trust-maintenance component so that it can maintain good spam coverage and a large pool of trustworthy users. The effectiveness of a reporter-based reputation system for spam filtering relies on an extensive preliminary study of the parameters of the system in the context of the ESP that will use it. An important parameter turns out to be the size of the reliable userset so that the system has a high spam coverage. It is also necessary to choose a low trust upgrade factor and a high trust downgrade factor in order to keep malicious users out of the reliable userset. The system may not be effective for ESP's for which the majority of the users are malicious.

Our work is the first comprehensive study on the practical effectiveness of a reporter-based reputation system. Our experiments were for a snapshot of the spam-filtering system and in that context, we showed that even a smaller userset can provide a good coverage of the spam that has not been caught by ESP's spam filters. We have shown that when the system is in production, and there may be malicious users trying to game it, the malicious users will not be able to have a significant detrimental effect. Possible future research directions include combining sender and reporter reputations in spam filtering and comparing different signature-generation algorithms for reputation-based spam filtering.

REFERENCES

Broder, A. 1997. On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of Sequences (SEQS)*. IEEE Computer Society, Italy, 21–29.

Chowdhury, A., Frieder, O., Grossman, D. A., and McCabe, M. C. 2002. Collection statistics for fast duplicate document detection. *ACM Transactions on Information Systems 20,* 2, 171–

191.

Cormack, G. and Bratko, A. 2006. Batch and online spam filter comparison. In *Proceedings of the Third Conference on Email and Anti-Spam*. Mountain View, CA, USA.

Dalvi, N., Domingos, P., Mausam, Sanghai, S., and Verma, D. 2004. Adversarial classification. In *Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining*. ACM, Seattle, WA, USA, 99–108.

DCC. 2006. Dcc reputations. http://www.rhyolite.com/anti-spam/dcc/reputations.html. Last accessed in August, 2006.

Dredze, M., Gevaryahu, R., and Elias-Bachrach, A. 2007. Learning fast classifiers for image spam. In *Proceedings of the Fourth Conference on Email and Anti-Spam*. Mountain View, CA, USA.

Drucker, H., Wu, D., and Vapnik, V. 1999. Support Vector Machines for Spam Categorization. *IEEE Transactions on Neural Networks 10,* 5, 1048–1054.

Fawcett, T. 2003. "In vivo" spam filtering: A challenge problem for data mining. *KDD Explorations 5,* 2, 203–231.

FTC. 2003. The can-spam act: Requirements for commercial emailers. http://www.ftc.gov/bcp/conline/pubs/buspubs/canspam.shtm. Last accessed in June, 2007.

Golbeck, J. and Hendler, J. 2004. Reputation network analysis for email filtering. In *Proceedings of the First Conference on Email and Anti-Spam*. Mountain View, CA, USA.

Goodman, J. and Yih, W. 2006. Online discriminative spam filter training. In *Proceedings of the Third Conference on Email and Anti-Spam*. Mountain View, CA, USA.

Hall, R. J. 1999. A countermeasure to duplicate-detecting anti-spam techniques. Tech. Rep. 99.9.1, AT&T Labs Research.

He, J. and Thiesson, B. 2007. Asymmetric gradient boosting with application to spam filtering. In *Proceedings of the Fourth Conference on Email and Anti-Spam*. Mountain View, CA, USA.

Henzinger, M. 2006. Finding near-duplicate web pages: A large-scale evaluation of algorithms. In *Proceedings of the Twenty Ninth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, Seattle, WA, 284–291.

Hovold, J. 2005. Naive bayes spam filtering using word-position-based attributes. In *Proceedings of the Second Conference on Email and Anti-Spam*. Mountain View, CA, USA.

Jonker, C. and Treur, J. 1999. Formal analysis of models for the dynamics of trust based on experiences. In *Proceedings of the of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW '99)*. Springer-Verlag, Spain, 221–231.

Kołcz, A. and Alspector, J. 2001. SVM-based filtering of e-mail spam with content-

specific misclassification costs. In *Proceedings of the IEEE ICDM Workshop on Text Mining (TextDM'2001)*. San Jose, CA.

Kołcz, A., Bond, M., and Sargent, J. 2006. The challenges of service-side personalized spam filtering: Scalability and beyond. In *Proceedings of the First International Conference on Scalable Information Systems (INFOSCALE)*. ACM, Hong Kong, 21.

Kołcz, A., Chowdhury, A., and Alspector, J. 2004. The impact of feature selection on signature-driven spam detection. In *Proceedings of the First Conference on Email and Anti-Spam*. Mountain View, CA, USA.

Ludeman, P. and Libbey, M. 2006. Algorithmically determining store-and-forward mta relays using domainkeys. In *Proceedings of the Third Conference on Email and Anti-Spam*. Mountain View, CA, USA.

Metsis, V., Androutsopoulos, I., and Paliouras, G. 2006. Spam filtering with naive bayes - which naive bayes? In *Proceedings of the Third Conference on Email and Anti-Spam*. Mountain View, CA, USA.

Meyer, T. and Whateley, B. 2004. Spambayes: Effective open-source, bayesian based, email classification system. In *Proceedings of the First Conference on Email and Anti-Spam*. Mountain View, CA, USA.

Prakash, V. and O'Donnell, A. 2005. Fighting spam with reputation systems. *Social Computing 3,* 9 (Nov.), 36–41.

Prakash, V. and O'Donnell, A. 2007. A reputation-based approach for efficient filtration of spam. http://www.cloudmark.com/releases/docs/wp_reputation_filtration_10640406.pdf. Last accessed in June, 2007.

Prince, M., Dahl, B., Holloway, L., Keller, A., and Langheinrich, E. 2005. Understanding how spammers steal your e-mail address: An analysis of the first six months of data from project honey pot. In *Proceedings of the Second Conference on Email and Anti-Spam*. Mountain View, CA, USA.

Ramchurn, S., Hyunh, T., and Jennings, N. 2004. Trust in multi-agent systems. *The Knowledge Engineering Review 19,* 1 (Mar.), 1–25.

Resnick, P. and Zeckhauser, R. 2002. Trust among strangers in internet transactions: Empirical analysis of ebays reputation system. *Advances in Applied Microeconomics 11*, 127–157.

Resnick, P., Zeckhauser, R., Friedman, R., and Kuwabara, E. 2000. Reputation systems. *Communications of the ACM 43,* 12, 45–48.

Rios, G. and Zha, H. 2004. Exploring support vector machines and random forests for spam detection. In *Proceedings of the First Conference on Email and Anti-Spam*. Mountain View, CA, USA.

Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. 1998. A Bayesian Approach to

Filtering Junk E-Mail. In *Proceedings of the AAAI-98 Workshop on Learning for Text Categorization*. Madison, WI.

Sarmenta, L. 2001. Volunteer computing. Ph.D. thesis, MIT.

Sculley, D. and Wachman, G. 2007. Relaxed online support vector machines for spam filtering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, Amsterdam, Netherlands, 415–422.

Symantec. 2004. White paper: Filtering technologies in symantec brightmail antispam 6.0. http://www.symantec.com/offer?a_id=19959. Last accessed in August, 2006.

Taylor, B. 2006. Sender reputation in a large webmail service. In *Proceedings of the Third Conference on Email and Anti-Spam*. Mountain View, CA, USA.

Witkowski, M., Artikis, A., and Pitt, J. 2001. Experiments in building experiential trust in a society of objective-trust based agents. *Trust in Cyber-societies Lecture Notes in Computer Science 2246*, 111–132.

Yih, W., Goodman, J., and Hulten, G. 2006. Learning at low false positive rates. In *Proceedings of the Third Conference on Email and Anti-Spam*. Mountain View, CA, USA.

Yoshida, K., Adachi, F., Washio, T., Motoda, H., Homma, T., Nakashima, A., Fujikawa, H., and Yamazaki, K. 2004. Density-based spam detector. In *Proceedings of KDD*. ACM, Japan, 486–493.