Filtering Spam with Behavioral Blacklisting

Anirudh Ramachandran, Nick Feamster and Santosh Vempala
College of Computing, Georgia Tech
{avr, feamster, vempala}@cc.gatech.edu

ABSTRACT

Spam filters often use the reputation of an IP address (or IP address range) as the basis for classification. Although this approach worked well when most spam originated from open relays and servers with fixed IP addresses, most spam today originates from IP addresses for which blacklist maintainers have outdated or inaccurate information (or no information at all). Spam campaigns are also spread out across a larger number of senders, reducing the amount of spam any particular IP address sends to a single domain, allowing spammers to stay “under the radar”. The dynamism and stealthiness of any particular IP address limits the accuracy and responsiveness of IP-based blacklists.

We present SpamTracker, a behavioral blacklisting algorithm that classifies email senders based on their sending behavior rather than identity; it cannot be deceived by spammers merely by using “fresh” IP addresses. SpamTracker uses fast clustering algorithms that react quickly to changes in sending patterns. We evaluate SpamTracker’s ability to classify spammers using email logs for over 150 email domains; we find that SpamTracker can correctly classify many spammers missed by current filtering techniques. SpamTracker is easy to replicate and distribute, and incorporating it into the existing email filtering infrastructure requires only small modifications to the configurations of existing mail servers.

1. Introduction

More than 75% of all email traffic on the Internet is spam [23]. To date, spam-blocking efforts have taken two main approaches: (1) content-based filtering and (2) IP-based blacklisting. Both of these techniques are losing their potency as spammers become more agile: To evade content-based filters, spammers have adopted techniques such as image spam and emails explicitly designed to mislead filters that “learn” certain keyword patterns. Spammers are now becoming increasingly successful in evading IP-based blacklists with nimble use of the IP address space (e.g., bots stealing IP addresses on the same local network, stealing IP address blocks with BGP route hijacking [27], etc.). To make matters worse, as most spam is now being launched by bots [27], spammers can send a large volume of spam in aggregate while only sending a small volume of spam to any single domain from a given IP address.

This “low and slow” spam sending pattern, combined with the facility with which spammers can quickly change the set of IP addresses from which they are sending spam, has rendered today’s methods of blacklisting spamming IP addresses far less effective than they once were. In particular, we note from our studies that, of the spam received at our spam “traps”, as much as 35% of spam was sent by IP addresses that were not listed by either Spamhaus [35] or SpamCop [34], two of the most reputable blacklists. Further, 20% of these IP addresses remained unlisted even after one month. Most of the IP addresses that were eventually blacklisted evaded the blacklist for about two weeks, and some evaded the blacklists for almost two months. Today’s blacklists cannot keep pace with the increasingly dynamic set of IP addresses that originate spam.

The ineffectiveness of these techniques stems from two shortcomings: First, existing blacklists are based on non-persistent identifiers, rather than behavior. An IP address does not suffice as a persistent identifier for a host: many hosts obtain IP addresses from dynamic address pools, which can cause aliasing both of hosts (i.e., a single host may assume different IP addresses over time) and of IP addresses (i.e., a single IP address may represent different hosts over time). Malicious hosts can spoof IP addresses and still complete TCP connections (by spoofing addresses on the same local area network or by hijacking BGP routes [27]), which allows spammers to introduce more dynamism. IP blacklists cannot keep up.

Second, information about email-sending behavior is compartmentalized by domain, not shared across domains. IP-based blacklists worked well when a small set of IP addresses (largely comprising open relays and dedicated servers) originated most spam, and each relay originated a large amount of spam. Today’s situation is different: a large fraction of spam now comes from botnets, large groups of compromised machines controlled by a single entity. With a much larger group of machines at their disposal, spammers now disperse their “jobs” so that each IP address sends spam at a low rate to any single domain. By doing so, spammers can remain below the radar, since no single domain may detect any single spamming IP address as suspicious. Detecting suspicious behavior now requires observing email sending patterns across domains.

Because of these shortcomings, IP-based blacklists are inaccurate and difficult to maintain. Maintainers of IP blacklists continually update their blacklists as a result of reports of spam campaigns mounted by armies of “fresh” IP addresses. Unfortunately, a spam campaign may complete by the time the IP addresses are blacklisted, at which time a new campaign with new IP addresses is imminent. While blacklisting all new IP addresses by default might constrain spammers, it also creates a nuisance for administrators when legitimate mail relays are renumbered, as well as for some mobile users that send email directly from their end systems. As a result, administrators must resort to heuristics to decide when and whether to remove IP addresses from a blacklist.

To address these shortcomings, we present behavioral blacklisting, which categorizes spammers based on how they send email, rather than the IP address (or address range) from which they are sending it. The intuition behind behavioral blacklisting is that, while IP addresses are ephemeral, spam campaigns, mailing lists, and techniques are more persistent. If we can identify emailing patterns that are characteristic of sending behavior, then we can continue to classify IP addresses as spammers, even as the IP addresses that spammers are using continue to change.

This paper presents the design, implementation, and evaluation of SpamTracker, a behavioral blacklisting algorithm that uses the set of target domains that a particular IP address sends mail to as the primary indicator of its behavior. We use the set of domains that an IP address targets within a fixed time window as the feature for clustering IP addresses that behave similarly. Our clustering algorithm takes as input an $n \times d \times t$ tensor, where $n$ is the number of IP addresses that sent email to any of $d$ domains within one
of $t$ time windows. The algorithm outputs clusters of IP addresses that exhibit similar sending patterns. Our evaluation of these clusters shows that spamming IP addresses form large clusters that are highly similar to each other but distinct from the behavior of IP addresses in other clusters. IP addresses of legitimate senders, on the other hand, do not form such clusters. SpamTracker can classify a “fresh” IP address as a spammer or a legitimate sender based on how closely its sending behavior (i.e., the set of domains that it targets) maps to a cluster that has been marked as known spamming behavior.

SpamTracker requires very little auxiliary information about whether an email sender is a spammer or a legitimate sender: it takes as input the email-sending patterns of all senders, builds clusters based on the sending behaviors of (a possibly small set of) known spammers, and classifies each sender based on whether its behavior is similar to a cluster that resembles known spamming behavior. Unlike conventional approaches which track individual IP addresses, SpamTracker tracks behavioral patterns to quickly identify whether a new IP address exhibits similar patterns to other previously seen IP addresses. Its ability to track behavior of groups, rather than individual IP addresses, allows it to adapt more quickly to ephemeral IP addresses that may not exhibit strong patterns from the perspective of any single domain.

Because SpamTracker classifies email based on sending behavior rather than on more malleable properties of email (e.g., content, or even IP address), we believe that spammers will have considerably more difficulty in evading SpamTracker’s classification methods. Nevertheless, SpamTracker must be agile enough to adapt to spammers’ changing behaviors: spamming patterns (i.e., which domains are targeted, and how they are targeted) will change over time, and adversaries that are aware of the SpamTracker algorithm may adjust their sending patterns to avoid falling into a particular cluster. We believe, however, that automated, large-scale behavior such as spamming will always give rise to clustering, and the challenge is to design SpamTracker to adapt the clusters it uses for classification, even as the spammers themselves attempt to evade them.

Using logs from an organization that manages email for over 150 domains, we find that SpamTracker detects many spammers before they are listed in any blacklist, indicating that SpamTracker complements today’s IP-based blacklists well.

The paper is organized as follows. Section 2 provides motivation for behavioral blacklisting. Section 4 presents a brief background on clustering techniques and details EigenCluster [6], the clustering algorithm that we use in SpamTracker. Section 5 describes the design and implementation of SpamTracker, and Section 6 presents our validation results and compares the performance of SpamTracker to state-of-the-art IP-based blacklists and spam trap deployments. In Section 7, we discuss various extensions of SpamTracker and deployment-related concerns. Section 8 presents related work, and Section 9 concludes.

2. Motivation

This section provides background on current email spamming practices and the performance of blacklists. In Section 2.1, we present the volumes and rates that each IP address in our traces sends to each domain; we find that spammers exhibit sending patterns that make it difficult to reliably detect and track spamming IP addresses. In Section 2.2, we provide background on current IP-based blacklisting techniques (e.g., DNS-based blacklists) and present a study of their effectiveness.

2.1 Spamming IPs: Ephemeral & Low-Volume

We present statistics on the network-level behavior of spammers, focusing on the techniques that make building the reputation of any particular IP address difficult. We study two aspects in particular:

- Persistence of spamming IP addresses. How much spam does a particular IP address send in a day, and how do the set of IP addresses change over time?
- Distribution of spamming IP addresses across target domains. What is the distribution of spam across target domains for any particular IP address, and how does this distribution change over time?

Lack of persistence in spamming IP addresses makes maintaining reputation about spammers based solely on IP addresses, since the blacklisted IP addresses keep changing. Similarly, an IP address that distributes spam evenly across target domains can help a spammer evade a blacklist entirely: maintenance of these lists typically requires explicit reports from a network about a “loud” IP address, so an IP address that is “low and slow” to any particular domain may be able to escape detection and blacklisting.

2.1.1 “New” IP addresses appear each day

IP-based blacklisting techniques are only effective if the IP addresses that send spam are stable over time (i.e., if the IPs that send spam do not change over time). To determine the extent to which spamming IP addresses remain stable over time, we study the IP addresses that send spam to over 150 distinct domains which collectively received 33 million pieces of spam during March 2007.1

Figure 1 shows the number of “new” IP addresses that these domains observed per day over the course of a month. The top line shows the fraction of IP addresses that were seen in the trace for a particular day that were never seen before in the trace (other lines show fraction of spam from IP addresses that appeared on the immediately preceding day, or within the month). Indeed, spam is not coming from the same IP addresses every day, and about 10% of IP addresses seen on any particular day were never seen before at any of the target domains. This finding suggests that, even given perfect mechanisms for maintaining reputation about email senders and relatively widespread observation, a significant number of IP addresses that have never been seen before are sending spam on any given day. Today’s reactive blacklists will not be able to stop spam these IP addresses, since they have no previous information about their activity.

2.1.2 Spamming IPs often target multiple domains

Existing blacklisting techniques collect reputation information about spam or spam senders based on the activity observed at a single domain (e.g., if a spammer sends a significant amount of spam to a single IP address, if it hits a spam trap, etc.) [34, 34]. Although many of these existing systems collect information from a large number of distributed domains, few, if any build reputation based on observed patterns across domains. We study the number of “victim” domains that each of these IP addresses targets, both over the course of a day and over the course of the month.

Previous work has shown that many bots that send spam are comparatively low-volume if observed at any one domain [27], but each of these IP addresses must send low volumes of spam to many domains for them to be “useful” to the spammer. Although only 84 of the domains we observed received any mail over a month (and the top 10 domains receive 90% of spam received by all 84 domains), Figure 2 shows that about half of all spam comes from about 15%
of IP addresses that target one or more domains, and about 35% of spam comes from IP addresses that target more than three or more domains. This result indicates that observing email sending patterns across domains could help expose sending patterns that are responsible for sending a significant amount of spam.

2.2 Blacklists: Incomplete & Unresponsive

After presenting a brief overview of blacklists and their most common operating mode (DNS-based blacklists, or “DNSBLs”), we briefly survey the performance of currently used DNS-based blacklists in terms of two metrics:

- **Completeness.** The fraction of spamming IP addresses (and fraction of spam from spammers) that is listed in a blacklist at the time the spam was received.
- **Responsiveness.** For the IP addresses eventually listed by a blacklist, the time for a blacklist to eventually list spamming IP addresses after they first send spam to any target domain.

Our results demonstrate that DNSBLs can be both incomplete and unresponsive. We present additional data that suggests that the email sending characteristics of spammers—in particular, their transience and the low volume of spam that they send to any domain in particular—limit the effectiveness of today’s blacklists.

2.2.1 Background: DNS-Based Blacklists (DNSBLs)

DNS-based Blacklists (DNSBLs) date back to 1997, with the Mail Abuse Prevention System (MAPS) [22]. MAPS was initially designed as a mechanism for blocking routes but was later released with a DNS-based interface. In a DNSBL, the maintainer keeps lists of IP addresses in a zone file. The DNSBL returns queries for any listed IP addresses with an IP address as a response code and answers queries for any unlisted IP address with an NXDOMAIN response.

DNSBLs offer a lightweight mechanism for querying a list of IP addresses, but the list membership must be maintained at least semi-manually. Maintenance of a blacklist entails not only deciding when a particular IP address should be added to a blacklist, but also when it should be removed. Blacklist maintainers typically add an IP address to a blacklist based on reports from network operators about an IP address that is sending an egregious amount of spam.

![Figure 1: Fraction of spamming IP addresses that were not observed at any of 150 domains for the preceding past 1 day, 1 month, and 2 months preceding.](image1)

<table>
<thead>
<tr>
<th>Source</th>
<th>Spammings IPs</th>
<th>Spammings from unlisted IPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trap 1</td>
<td>384,521</td>
<td>129,243 (35%)</td>
</tr>
<tr>
<td>Trap 2</td>
<td>172,143</td>
<td>64,386 (35%)</td>
</tr>
</tbody>
</table>

![Figure 2: Number of domains targeted by spamming IP addresses, and the amount of spam sent by these IP addresses, for a typical day’s worth of traffic.](image2)

Table 1: Fraction of spam at two spam traps from IP addresses that were unlisted in either Spamhaus or SpamCop, both at the time the message was received, and the fraction of spam from IPs that remained unlisted after 1 month.

(Which requires the spammer to raise the attention of an operator) or by sending spam to a particular spam trap or traps (which may not see the spam in the first place, particularly if spammers know to avoid them). Because reputation information about IP addresses can become “stale” (e.g., due to IP address dynamism, renumbering, etc.), the blacklist maintainer must determine how long an IP address should remain listed; this duration ranges from a 30 minutes to more than a year, depending on the nature of the problem and resolution.

2.2.2 Completeness

We study the completeness of “reactive” blacklists (i.e., those that only list IPs based on observed spamming activity or user reports as opposed to policy (e.g., SORBS [31]) lists all dynamic IP addresses irrespective of whether they were observed spamming or not. We consider the two most popular reactive blacklists, Spamhaus [35] and SpamCop [34]. To assess the completeness of existing DNSBLs, we first examine whether blacklists identify the spammers that we observe in one month of spam from two spam traps. We then observe received at a mail server that hosts email for hundreds of independent domains to determine how much of the mail that this provider accepted could have been blocked earlier if the provider had more complete blacklists at its disposal.

**Experiment 1: Are emails to spam traps blacklisted?** We first studied whether spammers were listed when they sent spam to two large spam traps. The two traps serve independent domains and they have no real email addresses, so we can consider all mail that these domains receive to be spam. Both run the MailAvenger [21]
SMTP server, which we have instrumented to measure whether a sender’s IP address is blacklisted at any of 8 blacklists at the time the email was received.

Trap 1 received 384,521 pieces of spam, of which 134,120 (35%) were received from IP addresses that were not listed in either Spamhaus or SpamCop when the spam was received. Trap 2 received 172,143 pieces of spam, of which 10% came from IP addresses that were not blacklisted. The significant fraction of spam coming from unlisted IP addresses suggests that IP-based blacklists are still relatively incomplete and that better blacklisting techniques could significantly reduce received spam. More troubling, perhaps, is that the blacklists remain incomplete even one month after each of these IP addresses sent spam: Unlisted IP addresses that accounted for 20% of spam at Trap 1, and 8.5% of spam at Trap 2, remained unlisted in Spamhaus blacklist one month after they were seen in our spam traps (see Table 1, suggesting that there is still a significant fraction of spam from senders that successfully evade conventional blacklisting techniques.

Experiment 2: Are accepted senders blacklisted later? The second set of logs are from an organization that hosts email service for over 700 domains, about 85 of which were active during March 2007. This provider’s mail servers reject or accept email based on a combination of techniques, including multiple blacklist lookups (Spamhaus [35], SORBS [31], etc.) and a large collection of customized heuristics. This provider blocks up to twice as much spam as any single blacklist.

Using our daily snapshot of the Spamhaus blacklist as a basis for comparison, we study the effectiveness of this email provider’s blocking heuristics by determining the fraction of mail that the provider accepts. Our results show that even this provider’s advanced filtering does not ensnare a significant collection of spammers: Of the 5,084,771 senders that passed the spam filters, 110,542 (2%) became listed in the Spamhaus blacklist during the following month. While a low fraction of all senders, we emphasize that the number of senders is still significant, that (from the previous section) Spamhaus may not list many spammers at all, and but, by its own estimates from user reports, about 15% of the remaining accepted email is spam.

2.2.3 Responsiveness

Many DNSBLs do not list an IP address before they receive multiple end-user reports about a spam sender; some even perform manual verification on top of this. Meticulous verification has the advantage of not accidentally blacklisting “good” senders, but doing so also limits responsiveness. In this section, we quantify the responsiveness of the Spamhaus DNSBL by determining, for the IP addresses that were eventually listed in April 2007, how long those IP addresses were active before they eventually were blacklisted.

As before, we use snapshots of the Spamhaus blacklist, but we also use hourly “difs” of the blacklist to determine when a new IP address is added. We examine email logs from March 1–31, 2007 and blacklist data from April 1–30, 2007. For each IP address that was not listed when it first sent spam to any of our spam sources but was eventually listed at some later point in April 2007, we compute the delay between the first occurrence of the IP at our trace to the first time that the IP address became listed in Spamhaus.³

³Because we only have the Spamhaus database for April, we cannot determine the exact listing time for IP addresses that were in the database on April 1, 2007; rather, we only know that they were listed between the time the spam was observed in March and April 1, 2007 (“less than 30 days” in Figure 3). If the IP address was not listed by April 1, 2007, we assume that whenever the IP becomes listed in April is the first time Spamhaus listed it.

Experiment 2: Are accepted senders blacklisted later? The second set of logs are from an organization that hosts email service for over 700 domains, about 85 of which were active during March 2007. This provider’s mail servers reject or accept email based on a combination of techniques, including multiple blacklist lookups (Spamhaus [35], SORBS [31], etc.) and a large collection of customized heuristics. This provider blocks up to twice as much spam as any single blacklist.

Using our daily snapshot of the Spamhaus blacklist as a basis for comparison, we study the effectiveness of this email provider’s blocking heuristics by determining the fraction of mail that the provider accepts. Our results show that even this provider’s advanced filtering does not ensnare a significant collection of spammers: Of the 5,084,771 senders that passed the spam filters, 110,542 (2%) became listed in the Spamhaus blacklist during the following month. While a low fraction of all senders, we emphasize that the number of senders is still significant, that (from the previous section) Spamhaus may not list many spammers at all, and but, by its own estimates from user reports, about 15% of the remaining accepted email is spam.

2.2.3 Responsiveness

Many DNSBLs do not list an IP address before they receive multiple end-user reports about a spam sender; some even perform manual verification on top of this. Meticulous verification has the advantage of not accidentally blacklisting “good” senders, but doing so also limits responsiveness. In this section, we quantify the responsiveness of the Spamhaus DNSBL by determining, for the IP addresses that were eventually listed in April 2007, how long those IP addresses were active before they eventually were blacklisted.

As before, we use snapshots of the Spamhaus blacklist, but we also use hourly “difs” of the blacklist to determine when a new IP address is added. We examine email logs from March 1–31, 2007 and blacklist data from April 1–30, 2007. For each IP address that was not listed when it first sent spam to any of our spam sources but was eventually listed at some later point in April 2007, we compute the delay between the first occurrence of the IP at our trace to the first time that the IP address became listed in Spamhaus.

Figure 3: Time-to-listing for the Spamhaus blacklist for IP addresses that were unlisted at the time spam was received, but were eventually blacklisted. Note: y-axis starts at 0.8.

Even when blacklists do list spamming IP addresses, the process of updating the blacklist may be quite slow. Figure 3 shows the time-to-listing for all IPs that were unlisted during the receipt of the email but eventually appeared at the blacklist in April 2007. In the case of the spam traps, 10–15% of spam senders that were unlisted at receipt of spam remained so 30 days after spam was received. The fraction is a strong indicator of the sluggishness of blacklists, because sending email to a spam trap automatically labels the sender as a spammer. In the case of the Organization that serves millions of real customers, almost 20% of senders that were unlisted when email was received but were eventually listed remain unlisted for over 30 days.

This analysis indicates that reactive blacklists are sometimes slow to respond, even for confirmed spammers; this slow responsiveness, coupled with the ability to continually send spam from “fresh” IP addresses (Section 2.1.1) represents a significant “window of opportunity” for spammers to send spam from non-blacklisted IPs. Motivated by this slow responsiveness, the next section proposes a complementary approach to blacklisting that is based on email sending patterns, rather than the reputation of an IP address alone.

3. The Case for Behavioral Blacklisting

The results from the previous section showed that spamming is both dynamic (i.e., from day-to-day, many new IP addresses are observed) and disperse (i.e., spammers are low-volume and can target multiple domains). Existing blacklisting techniques rely heavily on recognizing spamming behavior as observed at just a single domain (e.g., a significant volume of spam received at a domain that results in a blacklisting request, an IP address sending mail to a possibly well-known, avoidable spam trap, etc.). As a result, reactive blacklists are understandably incomplete and often slow to respond: by the time an IP address is blacklisted, the IP addresses from which most spam activity originates may have shifted. We expect that blacklists will become increasingly less effective as spammers further fine-tune their “low and slow” sending techniques so that individual IP addresses can escape notice.

Although individual IP addresses may change across time, we posit that (1) the sending patterns exhibited by spammers are sufficiently different from those of legitimate senders; and (2) those
patterns become more evident when email senders can be observed across many receiving domains. Based on these two hypotheses, the rest of the paper proposes a system called SpamTracker, which proactively blacklists email senders based on the set of domains they target. SpamTracker relies on a technique that we call behavioral blacklisting, which attempts to classify based on their network behavior, rather than their identity or the contents of the emails they send. While individual IP addresses may be ephemeral, they may exhibit “familiar” spamming patterns (i.e., similar to those of already well-known spamming IP addresses) that become evident when sending patterns are observed across multiple domains.

4. Clustering Algorithm

This section describes how we apply spectral clustering to design a behavioral blacklist system called SpamTracker. SpamTracker uses a spectral clustering algorithm proposed and analyzed by Kannan et al. [17] and made efficient in practice by Cheng et al. [6]. Section 4.1 presents an overview of the spectral clustering approach, and Section 4.2 describes how we apply spectral clustering to SpamTracker.

4.1 Spectral Clustering

Spectral clustering refers to partitioning algorithms that rely on the principal components of the input. There are generally two basic variants which can be viewed as (a) one-shot or (b) recursive. Given an object-feature matrix \( A \) with the goal of clustering the objects (rows) of \( A \), a one-shot algorithm would find the top few singular vectors of \( A \) (say \( k \)) and either project to their span or create a cluster for each one by assigning each row to that vector in which it has the largest component (a closely-related variant is to have two clusters per singular vector, one for rows with positive component in it and one for the ones with negative components). A recursive algorithm, on the other hand, uses one singular vector to partition the rows and recurses on the two parts. We focus on this type of algorithm.

The method in Cheng et al. [6] (“EigenCluster”) has two phases, a top-down divide phase followed by a bottom-up merge phase. In the divide phase, the algorithm normalizes a given nonnegative input matrix so that all rows have the same sum, then computes the second largest right singular vector. It sorts the rows according to their components in this vector and partitions this sequence at the point where the corresponding cut has minimum conductance (among the \( n-1 \) possible cuts of the sequence). The conductance of a partition is the total weight of entries across the partition divided by the smaller of the total weights incident to each side [17, 30]. After finding the partition, it recurses on each side (normalize, cut) till only singletons are left. This completes the divide phase whose end result can be viewed as a tree (the root represents all the rows, the leaves are individual rows). The merge phase finds a tree-respecting partition, i.e., one where every cluster corresponds to the entire subtree attached at some node of the tree. For a large class of objective functions, it does this by dynamic programming, in a bottom-up fashion. The specific function we use for the merge phase is called correlation clustering [6].

The next section describes the SpamTracker algorithm and how it incorporates clustering.

4.2 SpamTracker: Clustering Email Senders

SpamTracker classifies an email sender purely based on its sending behavior, ignoring content and variable handles for classification such as dynamically-allocated IP addresses. The intuition behind SpamTracker is that sending patterns of spamming hosts are similar to other senders and remain relatively stable, even as the IP addresses (or actual systems) that are sending the emails change. Consider the case of a spamming bot: Whatever the particular spamming behavior of a spamming bot, it is likely to be similar to other bots in its own botnet. Because botmasters in large botnets have only coarse-grained control over their bots, spamming patterns of bots will typically be similar across targeted domains even if each bot sends low volumes of spam to each domain. Thus, clustering spammers based on their sending patterns provides a way for their early detection, irrespective of their particular identities (e.g., the IP address) or blacklisting status. It follows from the above that, spam sent from even a newly-enlisted bot (i.e., from an IP address that has not been observed to send spam) will likely be caught by SpamTracker because its behavior will cluster it with other known bots in the botnet.

The SpamTracker algorithm uses semi-supervised learning and proceeds in two stages: (1) clustering and (2) classification. In the unsupervised clustering stage, SpamTracker accepts as input a \( n \times d \times t \) tensor \( M \), where \( n \) is the number of IP addresses that sent email to any of \( d \) domains within any of \( t \) particular time windows. Thus, \( M(i,j,k) \) denotes the number of times IP address \( i \) sent email to domain \( j \) in time slot \( k \). SpamTracker first collapses the time axis to obtain an \( n \times d \) matrix \( M' \), i.e.,

\[
M'(i,j) = \sum_{k=1}^{t} M(i,j,k).
\]

It clusters the matrix \( M' \) using the spectral clustering algorithm described in Section 4.1. The output of the clustering stage is the set of clusters of IP addresses \( C = C_1, C_2, \ldots, C_k \), where \( \cup_{i=1}^{k} C_i = \text{IPs in } M \) and \( C_i \cap C_j = \emptyset \) for \( i \neq j \). Logically, the set \( C \) consists of groups of IPs in \( M \) that have similar behavior in their target domains. Each cluster is associated with a traffic pattern, obtained by averaging the rows corresponding to IPs that fall in the cluster. For a cluster \( c \), we call this vector \( c_{\text{avg}} \).

\[
c_{\text{avg}} = \frac{\sum_{i=1}^{|c|} M(c,i)}{|c|}
\]

In the classification stage, SpamTracker accepts a \( 1 \times d \) vector \( r \) that corresponds to the recent behavior of an IP. It then calculates a score \( S(r) \) for this queried IP address using the following equation.
A known spamming cluster. Section 5.2 describes how SpamTracker "behavioral fingerprint" and determines whether this fingerprint resembles forms the front-end of lookups from mail servers and assigns scores to the queried senders back-end of SpamTracker.

5.1 Overview

In order to classify a new IP address when it arrives, we need to compute the clusters that form the basis of the classifier; and (2) classifying a new IP address as a likely spammer.

5. SpamTracker: Design and Implementation

This section describes how SpamTracker can be integrated into an existing email infrastructure. We present a brief overview of the system and then describe in detail its two basic operations: (1) computing the clusters that form the basis of the classifier; and (2) classifying a new IP address as it arrives.

5.1 Overview

The behavioral classifier that accepts queries for IP addresses and returns a score, \( S(r) \), for the IP's behavior.

\[
sim(r, c) = \frac{r \cdot c_{avg}}{|c_{avg}|}
\]

Intuitively, \( \sim(r, c) \) measures the similarity of the row vector \( r \) to cluster \( c \) by performing an inner product of \( r \) with the normalized average of rows in cluster \( c \). A cluster which has a similar set of target domains as \( r \) would have a large inner product.

We calculate the spam score \( S(r) \) as the maximum similarity of \( r \) with any of the clusters.

\[
S(r) = \max_c \sim(r, c).
\]

\( S \) can be used to filter or greylist spam by a mail service provider at or before the SMTP dialogue stage. We set a threshold such that if the row for an IP that is looked up has score higher than the threshold, it is flagged as spam. The threshold can be different for each cluster.

Querying an IP address is inexpensive: only Equations 1 and 2 need to be computed per lookup. The next section explains in detail how we implement SpamTracker, and the optimizations we use to improve the lookup speed and the overall robustness of the system.

5.2 Building the Classifier: Clustering

This section describes how SpamTracker accepts information about email senders as an \( IP \times \text{domain} \times \text{time} \) tensor and computes clusters of related senders (and corresponding average vectors). The classification component accepts queries for IP addresses and returns a score, \( S(r) \), for the IP's behavior.

As mentioned in Section 4.1, SpamTracker's clustering algorithms rely on the assumption that the set of domains that each spammer targets is often more stable than the IP addresses of machines that the spammer uses to send the mail. IP addresses of email senders may be highly dynamic due to various factors, including inadvertent or malicious re-addressing of machines in existing "spam armies", dynamic addressing, route hijacking, compromise of new botnets, etc. On the other hand, the set of domains that a particular address targets is often less malleable because it is determined by the spamming method used, membership in a particular botnet, the email mailing lists themselves, etc.

Accordingly, rather than maintaining reputations of senders according to their IP addresses, SpamTracker uses the vector representing how a sender sends traffic across domains, \( r \), as a "behavioral fingerprint" and determines whether this fingerprint resembles a known spamming cluster. Section 5.2 describes how SpamTracker builds clusters of known spammers, and Section 5.3 explains how SpamTracker determines whether an email sender's sending patterns resemble one of these clusters.

5.2 Building the Classifier: Clustering

SpamTracker uses the spectral clustering algorithm in Section 4.1 to construct the initial set of clusters. SpamTracker's clustering takes as input email sending patterns about confirmed spammers (i.e., the volume of email that each confirmed spamming IP address sends across some set of domains) over some time window to construct the matrix \( M(i, j, k) \). This input requires two components: (1) an initial "seed list" of bad IP addresses; and (2) email sending patterns for those IP addresses. This section describes in turn how SpamTracker might be able to acquire this type of data.
Data about spamming IP addresses is easy to obtain, and SpamTracker could use any such initial list of IP addresses to “bootstrap” its initial clusters. For example, an Internet Service Provider (ISP) that uses conventional SpamAssassin [33] filters to filter spam could use that list of IP addresses as its initial spammer IP addresses to be used for the basis for clustering.

The emailing patterns of each of the spamming IP addresses (i.e., the set of domains to which they send email) is more difficult to obtain because it requires visibility into the emails that many domains have received. Our evaluation of SpamTracker in Section 6 uses an email hosting provider’s decisions about early mail rejects from hundreds of domains to compute these clusters, but, in practice, a system like SpamTracker could also likely gain access to such data.

To build the rows in $M$ for each spamming IP address, participating domains could submit IP addresses that they have confirmed to be spammers as they do with blacklists, but based on our findings of the “low and slow” sending patterns of spammers (Section 2), SpamTracker will be most effective if it maintains sending patterns across domains for as many IP addresses as possible and subsequently clusters based on some subset of those that are labelled as spam by at least one domain. Fortunately, SpamTracker could obtain these sending patterns from receiving mail servers’ queries to the classifier, at least from some subset of trusted domains. Specifically, a lookup for IP address $a$ from domain $d$ is a reasonable indicator that $a$ has sent email to $d$, so SpamTracker can build vectors for all such addresses $a$ and then later build the matrix $M$ from just those addresses that are confirmed to be spammers.

### 5.3 Classifying Email Senders

Given an $r$ for some IP address $a$, SpamTracker determines whether this fingerprint resembles a confirmed spamming pattern by computing the similarity of $r$ to the sending behavior of some known spamming cluster. To compute this fingerprint, SpamTracker computes the spam score $S(r)$ (Equation 2, Section 4.2). SpamTracker maintains a vector $r$ for each IP address $a$ as it receives reports about sending behavior of an email from the mail servers of participating domains: SpamTracker collects these sending patterns as mail servers from trusted participating domains perform lookups to SpamTracker on address $a$, using the same method for collecting these patterns for all IP addresses during the clustering phase (described in Section 5.2).

Given a vector $r$ and the set of all clusters generated during clustering, SpamTracker returns a spam score $S(r)$. SpamTracker can simply return $S(r)$ to the querying mail server, which can incorporate this score into its existing mail filtering rules. We emphasize two benefits of SpamTracker’s classification process. First, $S(r)$ can be computed using only an IP address’s $r$ vector and the $c_{avg}$ rows for the spam clusters, both of which can be replicated and distributed (providing robustness against attack, as well as load balance). Clustering requires $r$ vectors from as many IP as possible and thus can be replicated; even though it requires aggregating sending information from many sending domains (and, hence, from potentially many SpamTracker replicas) this aggregation and clustering can be performed on a slower timescale than classification.

### 5.4 Tracking Changes in Sending Patterns

SpamTracker must recompute new clusters as sending patterns change. Our implementation of SpamTracker reclusters at fixed intervals, but in practice SpamTracker might only recluster when sending patterns no longer map to any existing clusters. Reclustering can be performed on a slower timescale than classification. Our implementation of SpamTracker reclusters at fixed intervals, but in practice SpamTracker might only recluster when sending patterns no longer map to any existing clusters. Reclustering can be performed on a slower timescale than classification.

#### 6. Evaluation

This section describes the evaluation of SpamTracker. In a real deployment, SpamTracker could compute clusters based on sending patterns across many domains for some time interval. To emulate this scenario, we construct the SpamTracker classifier by constructing $M(i, j, k)$ from the email logs of a large organization that manages mail servers for hundreds of domains. We use the matrix for time window $[t, t + \Delta t]$ to build the classifier, and the data in the window $[t + \Delta t, t + 2 \Delta t]$ to validate our classification. Section 6.1 summarizes the data sets used in our evaluation. Section 6.2 describes the properties of the resulting clusters and the validation results, and Section 6.3 describes our evaluation of SpamTracker’s ability to improve upon existing blacklisting and blocking techniques.

#### 6.1 Data

Table 2 summarizes the traces, their duration, and the data fields each trace provides. Our primary data is a set of email logs from many email servers of an organization (“Organization”) that hosts and manages mail servers for over 150 domains. The trace also contains an indication of whether it rejected the SMTP connection or not. We also use the full database of the Spamhaus [35] for one month, including all additions that happened within the month (“Blacklist”), to help us evaluate the performance of SpamTracker relative to existing blacklists. We choose the Blacklist traces for the time period immediately after the email traces end so that we can discover the first time an IP address, unlisted at the time email from it observed in the Organization trace, was added to Blacklist trace.

<table>
<thead>
<tr>
<th>Trace</th>
<th>Date Range</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization</td>
<td>Mar. 1 – 31, 2007</td>
<td>Received time, remote IP, targeted domain, whether rejected</td>
</tr>
<tr>
<td>Blacklist</td>
<td>Apr. 1 – 30, 2007</td>
<td>IP address (or range), time of listing</td>
</tr>
</tbody>
</table>

### 6.2 Clustering and Classification

To study the properties of the clusters that SpamTracker computes, we build the SpamTracker classifier using data for a window $\Delta t$ at time $t$, and use it to assign a spam score $S(r)$ to the sending information from many sending domains (and, hence, from potentially many SpamTracker replicas) this aggregation and clustering can be performed on a slower timescale than classification.

#### 6.2.1 Clustering and Classification

To study the properties of the clusters that SpamTracker computes, we build the SpamTracker classifier using data for a window $\Delta t$ at time $t$, and use it to assign a spam score $S(r)$ to the sending information from many sending domains (and, hence, from potentially many SpamTracker replicas) this aggregation and clustering can be performed on a slower timescale than classification.
reflect IP addresses whose sending patterns are very similar to the average rows of the classifier. As expected, the distribution of mails rejected by the organization tend towards larger values of \( S(r) \). We suspect that because legitimate email senders likely will not mimic each other’s sending patterns, the IP addresses in this region—both in the “accepted” and “rejected” plots—are likely to contain spammers. Indeed, in Section 6.3, we show that \textit{SpamTracker} correctly classified IP addresses in that were accepted by Organization but were eventually blacklisted.

Ideally, users of \textit{SpamTracker} should be able to set a single threshold for \( S(r) \) that clearly separates the majority of legitimate email from the majority of spam, but setting a single threshold for the experiment shown in Figure 7 could result in misclassifying a large fraction of received mail. For example, though setting a threshold of 10 would blacklist only about 5% of Organization’s accepted mail, it would only correctly classify 10% of all of the rejected mail. In fact, a lower threshold may be more appropriate: as we describe in Section 6.3 below, a significant fraction of accepted mail is still spam, and, in may cases \textit{SpamTracker} captures this spam before the Organization or Spamhaus does. Without ground truth data, it is difficult to determine a precise false positive rate, because “accepted” mail may simply be misclassified spam.

We believe that the quality of data (rather than the classification algorithm itself) is affecting our ability to separate the accepted and rejected mail with a single spam score. First, the data set is not cleanly labelled: the decisions of the Organization concerning whether to accept or reject a mail are not in fact a ground truth indicator as to whether mail is legitimate: The Organization estimates that as high as 15% of accepted mail is spam, and, as we show in Section 6.3, the emails that were accepted by the Organization for which \textit{SpamTracker} assigns high scores may in fact be undetected spammers. Second, \textit{SpamTracker} performs best when the representative sending behavior for a cluster is distributed across multiple domains, rather than concentrated in a single domain. Figure 7 shows that many emails have a spam score of 1, which implies that the classified IP address’s pattern is similar to a cluster whose average row is dominant in one column. According to Equations 1 and 2, this pattern will return a similarity of about \(|r|\). Because, in our dataset, a majority of senders in most small time windows send email to only a single domain, \(|r|\) is 1 for 50% of accepted email and 30% of rejected email. Our dataset does often has email senders that send mail to only a single domain in a time window.

In contrast, Figure 8 shows the distribution of \( S(r) \) for IP addresses that have maximum similarity with a single cluster whose \( c_{\text{avg}} \) not dominated by a single column. The “accepted” and “rejected” distributions are better separated because legitimate IP addresses that have maximum similarity with this cluster will likely not have hit all domains comprising the average row of this cluster (although the spammers in this cluster will likely hit all or most domains). As we discuss in more detail Section 7.4, better distribution of monitors could result in a more even observation of sending patterns which we believe would result in a distribution of \( S(r) \) that more closely resembles that shown in Figure 8.

### 6.3 Detecting Unknown Spammers

Because we know that 15% of accepted email in the trace is later reported as spam, we select (from Figure 7) the score corresponding to the highest-scoring 15% of legitimate email (i.e., the fraction of legitimate email most likely to be spam), \( S(r) = 5 \). We study all IP addresses for which the Organization accepted mail but were eventually blacklisted by Spamhaus. Of the 620 IP addresses that appeared in the Organization trace in the 6-hour window we analyzed, we observed 65 IPs (10%) for which \( S(r) > 5 \). In these cases, \textit{SpamTracker} could have labeled the senders as spammers before the Organization (or any blacklist) detected them.

### 7. Deployment Issues and Strategies

In this section, we survey how we envision how filtering techniques based on behavioral blacklisting could ultimately be deployed in an operational network, and our ongoing efforts to do so. We discuss how behavioral blacklisting scores might be integrated into existing email filtering systems, and some of the issues that may arise in building a system around \textit{SpamTracker} algorithms.

#### 7.1 Incorporation with Existing Systems

The results in Section 6 showed that \textit{SpamTracker} can classify spamming IP addresses in ways that are complementary to more conventional IP-based blacklisting, but an important question concerns how \textit{SpamTracker} can be incorporated to complement the ex-
existing deployments of mail servers and spam filters. We envision two possibilities:

**Option 1: Integration with existing infrastructure.** SpamTracker could be incorporated into existing mail filtering systems on mail servers by providing an additional “confidence score” for these filters that help them determine whether a particular piece of email is spam. Because SpamTracker provides a simple interface (i.e., it takes as input an IP address and returns a score), it can be incorporated into any existing spam filtering engine (e.g., SpamAssassin [33], MailAvenger [21]) in the same way that any other blacklist information would be added as a filtering criterion. A significant advantage to this technique is that deployment is simple and low-overhead: it requires adding a few lines of configuration to the mail server in the same way that any other DNSBL would work.

One disadvantage to this deployment approach is that it does not stop email traffic close to the source: the mail server that receives the spam drops the mail only after the traffic has already traversed the network and consumed resources on the receiving mail server.

**Option 2: “On the wire” deployment.** Unlike most existing spam filtering or classification systems, SpamTracker has the unique advantage that it can classify email senders solely based on the source IP address and destination domain of the mail being sent (i.e., it does not require examining or analyzing an email’s contents). Thus, another possibility for deploying SpamTracker involves deploying a network element that can examine traffic “on the wire” and identify connections to mail servers from IP addresses that fall into clusters with high spam scores. This deployment has the advantage that such a box could be deployed anywhere in the network, not just at the receiving mail server.

The disadvantage to this strategy is that deployment involves several additional steps: in particular, such a filtering element would need a channel to receive up-to-date information about both the email sending clusters (i.e., their average vectors, and their spamminess) and the vector for any particular sending IP address (i.e., to which domains it has sent). Maintaining up-to-date information about clusters and sending IP addresses in such a distributed, dynamic setting may prove challenging in practice.

### 7.2 Distributing SpamTracker

SpamTracker depends on statistics about spam sending patterns across multiple independent domains, which in turn implies the need for a system that can aggregate this data, compute the clusters, and return scores from many sources. The system must scale to a large number of contributors and must also be resistant to denial-of-service attacks.

To provide this type of support, we envision a federated mechanism for performing lookup (i.e., classification of senders) and a centralized one for performing the clustering itself. After the clusters themselves are computed, the average vectors for each cluster (i.e., the output of the algorithm) can be disseminated to many distributed points for the purposes of lookup: although data collection and clustering must be inherently centralized, recipients of email can perform classification by querying a system that is inherently distributed. This distributed system could then periodically pass all information about email sending patterns back to the clustering engine (i.e., on the timescale which clustering can be performed).

### 7.3 Defending Against Evasion and Attacks

Because SpamTracker depends on inputs from many distributed domains regarding the volume and times when spam arrives at their respective domains, it must provide be able to provide a system that can incorporate feedback about emailing patterns from a diverse group of email senders. The SpamTracker clustering engine must be robust not only to denial of service attacks, but also attacks on the system that mislead the clustering engine in ways that can cause spam to be misclassified as legitimate email, and vice versa. To control against denial of service attacks, the nodes that perform the classification can be distributed, and the clustering operation itself (which forms the clusters and computes the average vectors) can be replicated across many clustering engines.

To control against untrusted inputs, SpamTracker could form clusters based on email sending patterns from a smaller number of trusted email recipients (e.g., a few hundred trusted domains), each of which communicates with the SpamTracker system using a secure channel (e.g., a previously distributed secret key). In other words, SpamTracker can serve a larger number of domains; while SpamTracker’s clustering benefits from more inputs about email senders, it can serve as a classifier for a much larger set of domains that it does not necessarily trust to provide data for forming the clusters themselves.

### 7.4 Sensor Placement

As noted in Section 6, SpamTracker’s ability to recognize email sending patterns across domains requires the ability to observe the email sending patterns across a large set of domains. In particular, we believe that one of the reasons that the spam score $S(r)$ is unable to better separate spammers from many legitimate senders (as illustrated in Figure 7 by the large fraction of email, both rejected and accepted, that has $S(r) \leq 1$) is because the spam that is sent to the domains that we are observing is highly skewed: Recall from Section 2.1.2 that 90% of the spam we observe is received by only 84 of the 150 domains from which we observe email, and that only about 15% of the senders in our traces target more than one of the domains from which we can observe sending patterns at the email hosting provider. Based on our experiments using only clusters where the average vectors are less “skewed” towards a single domain (Figure 8), we expect that a broader distribution of senders across domains that receive similar volumes of email would further improve the SpamTracker classifier. We expect that many commercial spam filtering companies (e.g., IronPort [15], Secure Computing [28]) already have this data.

### 8. Related Work

In this section, we discuss several areas of related work. We first present previous characterization studies, several of which offer statistics that help build the case for behavioral blacklisting. We then survey existing systems for spam filtering, several of which use distributed monitoring but incorporate different algorithms for classification. Finally, we focus on various previous approaches for classifying email into legitimate email and spam, including both current and next-generation methods for generating blacklists.

**Blacklisting and identity.** SpamTracker relates closely to previous blacklisting proposals. Conventional blacklists essentially constitute lists of IP addresses of likely spammers (or sources of other unwanted traffic) and are intended to help spam filters [14, 21, 33] make better decisions about whether to block a piece of email based on the sender. Some blacklists are policy-based (e.g., they list all IP addresses that belong to a certain class, such as dialup addresses [31]). Other IP-based blacklists are “reactive”: they attempt to keep track of whether an IP address is a spammer, bot, phisher, etc. and keep this list up-to-date as hosts are renumbered, botnets move, and so forth [22, 34, 35, 37]. These blacklists essentially maintain lists of IP addresses and must be vigilantly maintained so as to not going out of date. The Spam Prevention Early Warn-
ing System (SPEWS) maintained a blacklist whereby senders could submit complaints about IP addresses that sent spam [32]. Submission of complaints was anonymous, and if spam continued to be sent from an IP address, SPEWS would proactively increase the range of IP address space that was blacklisted. It also kept track of addresses belonging to a spam facilitator (e.g., a web site hosting the spammers domain) in an attempt to predictively blacklist certain IP addresses.

SPF attempts to prevent IP addresses from sending mail on behalf of a domain for which they are not authorized to send mail [40], and domain keys associate a responsible identity with each mail [2]). Although both of these frameworks make it more difficult for an arbitrary IP address to send mail (thereby reducing each mail [2]), although both of these frameworks make it more difficult for an arbitrary IP address to send mail (thereby reducing each mail [40], and domain keys associate a responsible identity with each mail [2]). Although both of these frameworks make it more difficult for an arbitrary IP address to send mail (thereby reducing each mail [2]).

Collaborative filtering and whitelisting. SpamTracker is a collaborative blacklist: it takes inputs about sending behavior from many distributed domains to construct sending patterns. In this sense, SpamTracker resembles the many existing systems that take inputs from many distributed sources to build information about known spam (or spammers). Some of the most widely deployed collaborative filtering systems characterize known spam based on the contents of a piece of spam that was reported or submitted by another user or mail server [9, 10, 18, 24, 25, 38]. These systems allow mail servers to compare the contents of an arriving piece of email to the contents of an email that has been confirmed as spam; they do not incorporate any information about network-level behavior.

Various other systems and products collect information from distributed sets of users either to help filter spam or decrease the probability that legitimate mail is mistakenly filtered. Products from IronPort [16] and Secure Computing [29] sell spam filtering appliances to domains which then pass information about both legitimate mail and spam back to a central processing engine that in term improves the filters. The widespread deployment of these products and systems make them ideal candidates for the deployment of an algorithm like SpamTracker. “Re:” is a collaborative email whitelisting strategy that uses friend-of-friend relationships to determine whether or not an email sender is a likely spammer [13].

Characterization studies. A recent characterization study of the network-level behavior of spammers by Ramachandran and Feamster observes spamming behavior from the perspective of a single spam “trap” domain [27]. In this study, the authors observe that any particular IP address sends only a small volume of spam to the particular domain being observed over the course of 18 months. Duan et al. recently performed a similar study that observes similar characteristics [11]. Our characterization of spammers in Section 2 builds on these previous studies by observing email sending patterns across domains and time.

Content-independent blocking. SpamTracker complements existing content-based spam filtering systems by attempting to classify spam based on network traffic patterns, rather than on the contents of the email itself. Clayton’s “spamHINTS” project also aims to characterize and classify spam with analysis of network traffic patterns, rather than email contents [36]. More recent work on “extraction detection” involves instrumenting a mail server with a log processing program to detect senders of spam both at the local ISP [7] and in remote ISPs [8]. Although Clayton’s proposed methods are similar in spirit to our work (in that the methods rely on examining traffic patterns to distinguish legitimate email senders from spammers), the methods generally involve heuristics related to SMTP sessions from a single sender (e.g., variations in HELO messages, attempt to contact incoming mail servers to send outgoing mail); in contrast, SpamTracker relies on a wider deployment of traffic monitors (i.e., it relies on observing email sending patterns from many domains) but is then able to for more protocol agnostic “fingerprints” for email senders that are likely spammers.

Trinity is a spam filtering system built on top of a distributed database that attempts to use content-independent methods to identify emails that are likely sent from bots [5]. Trinity’s heuristics for content-independent filtering rely simply on detecting IP addresses that originate a large number of emails in a short period of time; it uses a distributed database to keep track of these email sending rates. Because SpamTracker also relies on counting the volume of emails that each sender sends across domains, we envision that the distributed database proposed for Trinity could also be used to track email sending patterns for input to SpamTracker.

Clustering for spam classification. Previous studies have attempted to cluster spammers based on an email contents, such as the URLs contained in the bodies of the emails [3, 20]. Li et al. focus on clustering spam senders to predict whether a known spammer will send spam in the future [20], and Anderson et al. cluster spam according to URLs to better understand the relationship between the senders spam messages that advertise phishing and scam sites and the Web servers that host the scams themselves [3]. These systems cluster emails based on content, while SpamTracker clusters email senders based on their sending behavior. Unlike the methods of Li et al., SpamTracker’s clustering techniques can also classifying previously unseen IP addresses.

Throttling outgoing spam. SpamTracker focuses on filtering spam after it has been sent, and complements previous proposals that have suggested throttling senders using various schemes such as stamps, proof-of-work, etc. One of the most common postage schemes is called “bankable postage”, whereby senders obtain stamps or tokens from some authority and then attach these tokens to emails [1, 39]. Other types of techniques for throttling spam require the sender to issue some “proof of work”, either in CPU memory [12], although these schemes have also been criticized because, in certain circumstances, they can prevent legitimate users from sending normal volumes of email [19].

9. Conclusion

This paper presented SpamTracker, a system that classifies email senders using a technique we call behavioral blacklisting. Rather than classifying email senders according to their IP addresses, behavioral blacklisting classifies senders based on their sending patterns. Behavioral blacklisting is based on the premise that many spammers exhibit similar, stable sending patterns that can act as “fingerprints” for spamming behavior.

SpamTracker is a behavioral blacklisting system that uses spectral clustering email senders based on the set of domains that they target. SpamTracker uses cross-domain sending patterns of confirmed spammers to build clusters, each of which has an average vector that represents a spamming pattern for that cluster. SpamTracker also tracks sending patterns of other email senders and computes the similarity of their sending patterns to that of a known spam cluster as the basis for a “spam score”. Our evaluation using email logs from an email provider that hosts over 150 independent domains shows that SpamTracker offer improvements over existing blacklists: we find that SpamTracker correctly classifies much mail that is known to be spam and also detects many spammers before they are listed in any blacklist. SpamTracker’s design makes it
easy to replicate and distribute, and deploying it requires only small modifications to the configurations of existing mail servers.

SpamTracker has demonstrated the promise of behavioral black-listing, and our ongoing work involves gathering data from a wider set of domains as well as deploying our system and incorporating it into existing filtering systems to allow us to evaluate SpamTracker in practice.

REFERENCES


[34] SpamCop. http://www.spamcop.net/.


