Filtering Spam with Behavioral Blacklisting

Anirudh Ramachandran, Nick Feamster, and Santosh Vempala
College of Computing, Georgia Tech
801 Atlantic Drive, Atlanta, GA - 30332, USA
{avr,feamster,vempala}@cc.gatech.edu

ABSTRACT

Spam filters often use the reputation of an IP address (or IP address range) to classify email senders. This approach worked well when most spam originated from senders with fixed IP addresses, but spam today is also sent from IP addresses for which blacklist maintainers have outdated or inaccurate information (or no information at all). Spam campaigns also involve many senders, reducing the amount of spam any particular IP address sends to a single domain; this method allows spammers to stay "under the radar". The dynamism of any particular IP address begs for blacklisting techniques that automatically adapt as the senders of spam change.

This paper presents *SpamTracker*, a spam filtering system that uses a new technique called *behavioral blacklisting* to classify email senders based on their sending *behavior* rather than their identity. Spammers cannot evade *SpamTracker* merely by using "fresh" IP addresses because blacklisting decisions are based on sending patterns, which tend to remain more invariant. *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 115 email domains; we find that *SpamTracker* can correctly classify many spammers missed by current filtering techniques. Although our current datasets prevent us from confirming *SpamTracker*'s ability to completely distinguish spammers from legitimate senders, our evaluation shows that *SpamTracker* can identify a significant fraction of spammers that current IP-based blacklists miss. *SpamTracker*'s ability to identify spammers before existing blacklists suggests that it can be used in conjunction with existing techniques (*e.g.*, as an input to greylisting). *SpamTracker* is inherently distributed and can be easily replicated; incorporating it into existing email filtering infrastructures requires only small modifications to mail server configurations.

Categories and Subject Descriptors: C.2.0 [Computer Communication Networks]: Security and protection

General Terms: Security, Design, Algorithms

Keywords: spam, botnets, blacklist, security, clustering

1. INTRODUCTION

More than 75% of all email traffic on the Internet is spam [25]. 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 also evading IP-based blacklists with nimble use of the IP address space (*e.g.*, stealing IP addresses on the same local network [19], stealing IP address blocks with BGP route hijacking [30]). To make matters worse, as most spam is now being launched by bots [30], 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 and the ease with which spammers can quickly change the IP addresses from which they are sending spam has rendered today's methods of blacklisting spamming IP addresses less effective than they once were [11]. For example, our study in Section 2 shows that, of the spam received at our spam "traps", as much as 35% was sent from IP addresses that were not listed by either Spamhaus [37] or SpamCop [36], two 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.

Two characteristics make it difficult for conventional blacklists to keep pace with spammers' dynamism. First, *existing blacklists are based on non-persistent identifiers.* 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 steal IP addresses and still complete TCP connections, which allows spammers to introduce more dynamism. IP blacklists cannot keep up. Second, *information about email-sending behavior is compartmentalized by domain and not analyzed across domains.* Today, a large fraction of spam 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 deem any single spamming IP address as suspicious.

IP blacklists must be continually updated to keep pace with 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. Blacklisting all new IP addresses is not an option, either: it creates a nuisance for administrators when legitimate mail relays are renumbered, as well as for some mobile users.

To keep pace with this dynamism, we propose a new technique called *behavioral blacklisting*, which complements existing blacklists by categorizing 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 as identifiers, spam campaigns, spam lists,

CCS'07, October 29–November 2, 2007, Alexandria, Virginia, USA.

and spamming techniques are more persistent. If we can identify email-sending patterns that are characteristic of spamming behavior, then we can continue to classify IP addresses as spammers even as spammers change their IP addresses.

We design 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 and incorporate this algorithm into a system called *SpamTracker*. 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 large 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. Using logs from an organization that manages email for over 115 domains, we find that *SpamTracker* detects many spammers *before they are listed in any blacklist*, suggesting that *SpamTracker* can complement today's IP-based blacklists by catching some spammers earlier than they would otherwise be caught.

SpamTracker requires 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.

The paper is organized as follows. Section 2 motivates behavioral blacklisting. Section 3 presents a brief background on clustering techniques and describes EigenCluster [7], the clustering algorithm that we use in *SpamTracker*. Section 4 describes the design and implementation of *SpamTracker*, and Section 5 presents our validation results and compares the performance of *SpamTracker* to state-of-the-art IP-based blacklists and spam trap deployments. In Section 6, we discuss various extensions of *SpamTracker* and deployment-related concerns. Section 7 presents related work, and Section 8 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 at which IP addresses in our traces send spam 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.

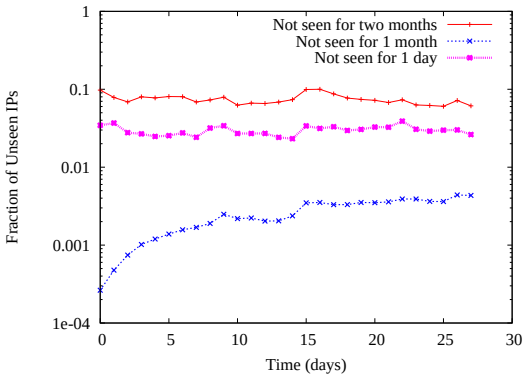


Figure 1: Fraction of spamming IP addresses that were not observed at any of 115 domains for the past 1 day, past month, and past 2 months.

2.1 The Behavior of Spamming IP Addresses

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: (1) *Persistence:* How much spam does a particular IP address send in a day, and how does the set of IP addresses change over time? (2) *Distribution:* What is the distribution of spam *across target domains* for any particular IP address, and how does this distribution change over time?

2.1.1 Persistence: "New" IP addresses every day

To determine the extent to which spamming IP addresses remain stable, we study the IP addresses that send spam to over 115 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 coming from different 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. Thus, 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.

Lack of persistence in spamming IP addresses makes maintaining reputation about spammers based solely on IP addresses difficult, since the blacklisted IP addresses keep changing. Given no previous information about the activity of an IP address, a conventional blacklist will not be able to reliably block spam from that address.

2.1.2 Distribution: Some IPs target many 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.) [36, 37]. Although

[1] Section 5.1 describes this dataset (as well as the others that we used in our evaluation) in more detail.

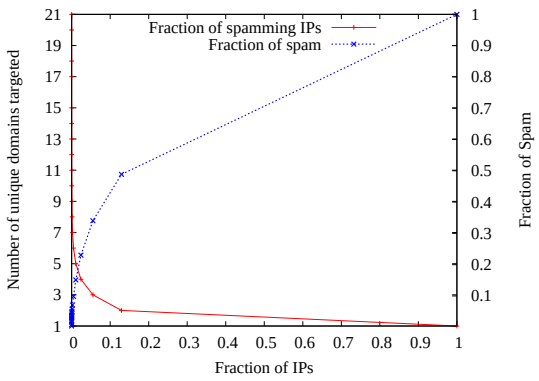


Figure 2: Fraction of spam sent (y' axis), and number of domains targeted (y axis), by spamming IP addresses for a typical day's worth of traffic at the email provider's servers. The IPs are reverse sorted by number of spam messages produced.

some existing systems collect information from a large number of distributed domains, few, if any, build reputation based on observed patterns *across* domains. Thus, an IP address that distributes spam evenly across target domains may 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.

Previous work has shown that many bots that send spam are comparatively low-volume if observed at any one domain [30], but each of these IP addresses must send low volumes of spam to *many* domains for them to be "useful" to the spammer. Our analysis confirms this conjecture: Figure 2 shows that about half of all spam (y' axis) comes from the top 15% spamming IP addresses (x-axis); this subset of IPs targets two or more domains (y-axis). Similarly, the top spamming IPs responsible for up to 35% of spam target three or more domains. Thus, *observing email sending patterns across domains could help expose sending patterns that are responsible for sending a significant amount of spam*.

2.2 The Performance of IP-Based Blacklists

After presenting a brief overview of IP-based 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 in response to dynamism in IP addresses. 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 single domain—make it difficult for blacklists to track the IP addresses of some spammers.

2.2.1 Background: DNS-Based Blacklists (DNSBLs)

DNSBLs are a "hack" on the DNS name resolution infrastructure to allow users to query for blacklisted IP addresses using existing DNS client and server protocols and utilities. A DNSBL maintainer keeps blacklisted IP addresses in a zone file; the server responds to a query for a listed IP address (encoded in a domain name) with another IP address (usually an address such as `127.0.0.2` that has no meaning in the DNS resolution infrastructure) but returns an `NXDOMAIN` for an unlisted address. DNSBLs offer a lightweight mechanism for querying a list of IP addresses, but *the list membership must be maintained at least semi-manually*. Maintenance 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 (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 30 minutes to more than a year, depending on the nature of the problem and resolution.

			Spam from unlisted IPs	
Data Source	*Spam*	*IPs*	*At Receipt*	*After 1 Month*
Trap 1	384,521	129,243	134,120 (35%)	79,532 (20%)
Trap 2	172,143	64,386	17,132 (10%)	14,534 (8.5%)

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.

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 [34]) lists all dynamic IP addresses irrespective of whether they were observed spamming or not). We consider the two most popular reactive blacklists, Spamhaus [37] (specifically the XBL and SBL zones) and SpamCop [36]. 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 mail received at a 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 during March 2007. 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.[2] Both run the MailAvenger [23] 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 *complementary* blacklisting techniques could significantly reduce spam. Additionally, *blacklists may 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 signifi-

[2] One of the domains serves eight legitimate users. We exclude this legitimate mail from our analysis and do not expect the presence of these addresses to have an effect on the spam received at the domain.

cant 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 (our observation period). This provider's mail servers reject or accept email based on a combination of techniques, including multiple blacklist lookups (Spamhaus [37], SORBS [34], 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, *only* 110,542 (2%) became listed in the Spamhaus blacklist during the following month. This fraction is significantly lower than the 15% quoted by this provider as the fraction of accepted email that is later reported as spam, which suggests that current blacklists remain incomplete, even after long periods of time.

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. Meticulous verification can reduce the likelihood of 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 "diffs" of the blacklist to determine when a new IP address was 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 one of our spam traps 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]

Even when blacklists *do* list spamming IP addresses, the process of updating the blacklist may be 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 provider that serves millions of real customers ("Organization"), almost 20% of senders that were unlisted when email was received *remain unlisted for over 30 days* before eventually appearing in the blacklist.

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]Because 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. This assumption is reasonable as Spamhaus lists *persistent* spammers for a minimum of 30 days [1].

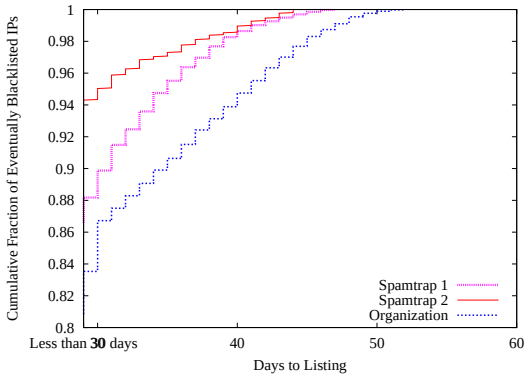


Figure 3: Time-to-listing for the Spamhaus blacklist for IP addresses that were unlisted at the time spam was received, but were eventually blacklisted. *Note:* y-axis starts at 0.8.

2.3 The Case for Behavioral Blacklisting

Although individual IP addresses' sending behavior 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.

3. CLUSTERING ALGORITHM

SpamTracker uses a spectral clustering algorithm proposed and analyzed by Kannan *et al.* [18] and made efficient in practice by Cheng *et al.* [7]. Section 3.1 presents an overview of the spectral clustering approach, and Section 3.2 describes how we apply spectral clustering within *SpamTracker*.

3.1 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 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.* [7] ("EigenCluster") has two phases: a top-down divide phase and 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

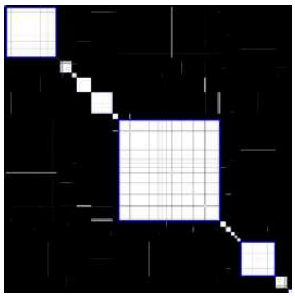


Figure 4: An IP$\times$ IP matrix of related spam senders; IP addresses that send mail to similar sets of domains are grouped into distinct clusters; the intensity of a pixel at (i,j) indicates i's similarity to j.

of the total weights incident to each side [18, 33]. After finding the partition, it recurses on each side until only singletons remain. This completes the divide phase, whose end result is 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 many 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* [7].

3.2 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 [26], 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 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':

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

It clusters the matrix M' using the spectral clustering algorithm described in Section 3.1. The output of the clustering stage is the set of clusters of IP addresses $C = C_1, C_2, \ldots, C_k$, where $\cup_{i=1}^{k} C_i =$ IPs in M and $C_i \cap C_j = \phi$ 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_{avg}.

$$c_{avg} = \frac{\sum_{i=1}^{|c|} M'_c(i)}{|c|}$$

where $M'_c(i)$ is the submatrix comprising the rows of cluster 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.

$$sim(r, c) = \frac{r \cdot c_{avg}}{|c_{avg}|} \quad (1)$$

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 that 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.

$$S(r) = \max_{c} sim(r, c). \quad (2)$$

S can be used to filter or *greylist* (*i.e.,* temporarily reject with the assumption that a legitimate mail sender will eventually retry) 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 the design of *SpamTracker* in detail and the optimizations we use to improve the lookup speed and the overall robustness of the system.

4. DESIGN

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 when it arrives.

4.1 Overview

The spectral clustering algorithm in Section 3.2 serves as the back-end of *SpamTracker*. The behavioral classifier that accepts lookups from mail servers and assigns scores to the queried senders forms the front-end. Figure 5 shows the high-level design of *SpamTracker* and the interaction between the back-end (which performs clustering and classification operations) and the interface to mail servers (which receives email sending patterns as input to the clustering algorithm and answers queries about the status of any particular IP address); to an ordinary mail server, the interface to *SpamTracker* looks like any other DNS-based blacklist, which has the advantage that existing mail servers need only to be reconfigured to incorporate *SpamTracker* into spam filtering decisions. We discuss how *SpamTracker* can be incorporated into existing infrastructure in Section 6.2.

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. 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 4.2 describes how *SpamTracker* builds clusters of known spammers, and Section 4.3 explains how *SpamTracker* determines whether an email sender's sending patterns resemble one of these clusters.

4.2 Clustering

SpamTracker uses the spectral clustering algorithm from Section 3.1 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 [35] 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 sending patterns of each of the spamming IP addresses is more difficult to obtain because it requires visibility into the emails that many domains have received. Our evaluation of *SpamTracker* (Section 5) uses an email hosting provider's decisions about early mail rejects from hundreds of domains to compute these clusters, but, in practice, other systems 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[4], at least from some subset of trusted domains.[5] 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 later build the matrix M from just those addresses that are confirmed to be spammers.

4.3 Classification

SpamTracker maintains a vector representing the sending pattern, r, for each IP address a compiled from reports 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 4.2).

Given an r for some IP address a, *SpamTracker* returns a score $S(r)$ (computed using Equation 2, Section 3.2) whose magnitude determines how closely this fingerprint resembles a confirmed spamming pattern (*i.e.*, cluster). *SpamTracker* can simply return $S(r)$ to the querying mail server, which can then incorporate this score into its existing mail filtering rules. An important benefit of the classification process is that $S(r)$ 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 addresses as possible; 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.

[4]Note that the query mechanism needs a way of finding the email domain name of the organization performing the query. DNS reverse lookups, or extra information in the query packets, could provide such a mechanism.

[5]Because previous work has observed that bots occasionally perform reconnaissance queries against blacklists [29], we cannot assume that *all* queries to the blacklist reflect the receipt of email by a mail server.

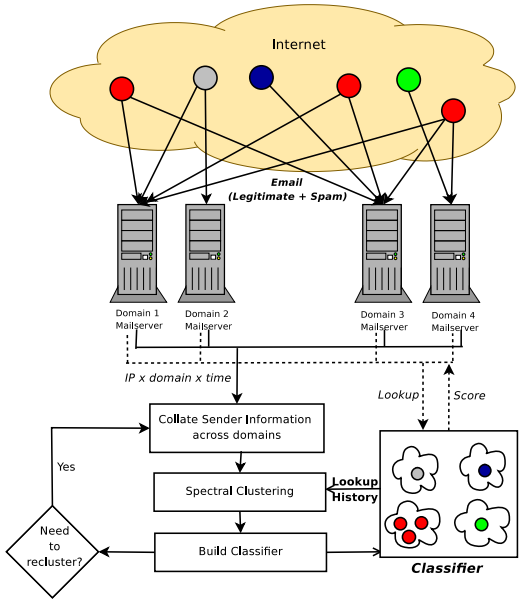


Figure 5: The high-level design of *SpamTracker*. The *clustering* component of *SpamTracker* accepts information about email senders as an $IP \times domain \times time$ tensor and computes clusters of related senders (and corresponding average vectors). The *classification* component accepts queries for IP addresses and returns a score, $S(r)$, for the IP's behavior.

4.4 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 cost (time, memory, CPU) increases with larger input matrices, so clustering on very large time windows may be impractical. We use an efficient re-clustering method that preserves historical information but keeps clustering cost approximately constant. At the beginning of each clustering phase, we add all average rows from the previous clustering stage scaled by the size of the cluster each row represents, which produces the effect of clustering on the input of both stages without the added cost.

5. 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 at $[t, t + \Delta t)$ to build the classifier, and the data in the window $[t + \Delta t, t + 2\ \Delta t)$ to validate our classification. Section 5.1 summarizes the data sets used in our evaluation. Section 5.2 describes the properties of the resulting clusters and the validation results, and Section 5.3 describes our evaluation of *SpamTracker*'s ability to improve upon existing blacklisting and blocking techniques by classifying spammers ahead of blacklists.

5.1 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

Trace	Date Range	Fields
Organization	Mar. 1 – 31, 2007	Received time, remote IP, targeted domain, whether rejected
Blacklist	Apr. 1 – 30, 2007	IP address (or range), time of listing

Table 2: Data sets used in evaluation.

a provider ("Organization") that hosts and manages mail servers for over 115 domains. The trace also contains an indication of whether it rejected the SMTP connection or not. We also use the full database of Spamhaus [37] 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.

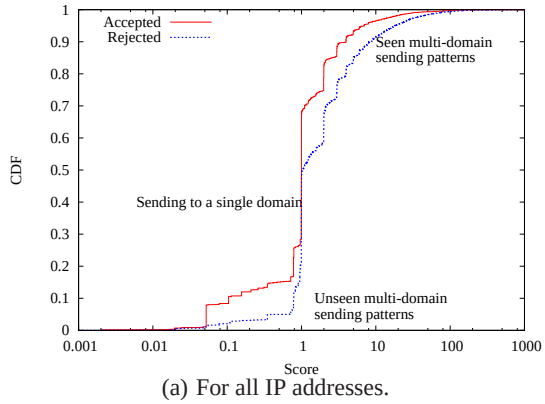
5.2 Clustering and Classification

To study the properties of the clusters that *SpamTracker* computes, we build the *SpamTracker* classifier using data for a window Δt at time t, and use it to assign a spam score $S(r)$ senders in the window $[t + \Delta t, t + 2\ \Delta t)$. We set Δt to be 6 hours; clustering using different time intervals (which we intend to explore in future work) may also help *SpamTracker* perform better.

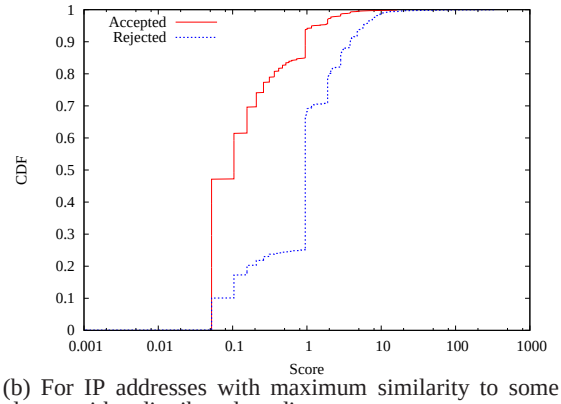
Figure 6(a) shows the distribution of these scores for all IP addresses in a 6-hour window, separated into two plots based on whether the Organization decided to reject the mail early or accept it for delivery. A high score implies that the sending pattern for the classified IP is similar to a known spamming pattern. The low-score region (where $S(r) < 1$) comprises IP addresses whose patterns are unknown to the classifier. Senders that map into this range should not necessarily be considered legitimate; rather they simply do not have a recognized, blacklisted sending pattern. High scores 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 5.3, we show that *SpamTracker* correctly classified IP addresses in that were accepted by the Organization but were eventually blacklisted.

Ideally, users of *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 6(a) could result in misclassifying a large fraction of received mail. For example, though setting a threshold of 10 would blacklist only about 5% of the 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 5.3 below, a significant fraction of accepted mail is still spam, and, in many cases, *SpamTracker* captures this spam before the Organization or Spamhaus does. However, *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 much as 15% of accepted mail is spam, and, as we show in Section 5.3, the emails that were accepted by the Organization for which *SpamTracker* assigns high scores may in fact be undetected spammers. Second, *SpamTracker* performs best when the representative sending behavior for a cluster is distributed across multiple domains, rather than concentrated in a single domain. Figure 6(a) 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 often has email senders that send mail to only a single domain in a time window.



(a) For all IP addresses.



(b) For IP addresses with maximum similarity to some cluster with a distributed sending pattern.

Figure 6: Score distribution for *SpamTracker*'s classification for (a) All IPs in a "round" of classification, and (b) IPs that have maximum similarity with a cluster whos c_{avg} is not dominated by a single column. Evaluation for a 6-hour period using a classifier trained using the previous 6-hour window.

Figure 6.5 shows the distribution of $S(r)$ for IP addresses that have maximum similarity with a single cluster whose c_{avg} is not dominated by a single column. The "accepted" and "rejected" distributions separate more cleanly because legitimate IP addresses that have maximum similarity with this cluster will likely not have sent mail to all domains comprising the average row of this cluster (although the spammers in this cluster will likely hit all or most domains). A better distribution of monitors might result in a more even observation of sending patterns, which should result in a distribution of $S(r)$ that more closely resembles that shown in Figure 6.5.

5.3 Detecting "New" Spammers

To estimate the fraction of spammers that *SpamTracker*'s clustering techniques can detect in advance of conventional blacklisting techniques, we study a subset of the email with the highest spam scores and determine whether any emails from this subset were eventually reported as spam (*e.g.*, by users, or some other auxiliary technique). Operators at the Organization acknowledge that

about 15% of email that is initially accepted falls into this category. To estimate how well *SpamTracker* would perform at catching this 15% of misclassified mail, we examine the 620 emails were initially missed by the Organization's filters but eventually blacklisted in the following month. Of these, 65 emails (about 10%) had a spam score of $S(r) > 5$ (from Figure 6(a)), suggesting that *SpamTracker* could complement existing filtering mechanisms by capturing additional spam that existing filters miss.

6. DISCUSSION

Although the general method of *behavioral blacklisting* shows promise for fast classification of spammers, there is much room for improvement, particularly with respect to the classification algorithms (which could, for example, incorporate other features as input). This section proposes specific areas where the classification algorithms could be improved, surveys how filtering techniques based on behavioral blacklisting could ultimately be deployed in an operational network, and presents our ongoing efforts to do so. We also discuss how behavioral blacklisting scores might be integrated into existing spam filtering systems and some of the issues that may arise in implementation and deployment.

6.1 Improving Classification

IP addresses that are most similar to a single spamming cluster can be classified more accurately. In order to achieve this separation for *all* new IPs, we propose two improvements to *SpamTracker* that may result in better clusters.

Using more features for clustering. Although *SpamTracker* uses target domains to construct the initial object-feature matrix (Section 3.2), other behavioral features may be able to better classify spammers. Temporal patterns such as the time interval between successive emails received from an IP (or alternatively, the sending frequency of the IP) is one such feature. Botmasters often manage all their bots using unified interfaces that may also be used to disseminate spam templates and mailing lists to bots [26], so these bots may exhibit similar temporal behavior (perhaps spamming frequencies) in addition to their similarity in target domains.

Improved similarity computation. In Equation 1, all columns of IP's "fingerprint" vector, r, are weighted equally. Some domains may be better at distinguishing one cluster of spammers from another. For example, spammers targeting victims in different countries may send email to country specific domains as well as to ubiquitous domains (*e.g.*, `gmail.com`). In this case, the country-specific domains may be more helpful in distinguishing the two sets of spammers. Our ongoing work includes experimenting with an algorithm that weights each column (domain) differently.

6.2 Incorporating with Existing Systems

We discuss how *SpamTracker* can be incorporated to complement the existing deployments of mail servers and spam filters. We describe two possibilities below: integration with existing filters and on the wire deployment. In either case, the back-end of *SpamTracker* can remain the same: it only needs to run a DNS server (or another popular query interface such as XML-RPC) that accepts requests for IP addresses, retrieves the classification score $S(r)$ from the *SpamTracker* classification engine, and returns the score to the client. In this sense, *SpamTracker* is a stand-alone system that can even be used internally within an organization.

Option 1: Integration with existing infrastructure. *SpamTracker* could be incorporated into existing 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 in terms of sender behavior. 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 [35], MailAvenger [23]) in the same way that any other blacklist information would be added as a filtering criterion. Using this system would be easy: the addition of one line to the configuration of most mail filtering software should allow users to benefit from *SpamTracker*'s filtering strategy.

The disadvantage, however, 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. Such a system 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.

6.3 Deployment Challenges

A *SpamTracker* deployment must be scalable (*i.e.*, it must be able to handle a large volume of email and a large number of senders) and robust (*i.e.*, it must be resistant to attack and remain highly available). To achieve these goals, we believe that *SpamTracker* could ultimately be distributed: many servers (possibly the same ones who manage mail for various domains) report sender behavior to a centralized location that performs the clustering. *SpamTracker* must aggregate data from many domains, compute the corresponding clusters of email senders, and return scores from many sources; in doing so, it faces scalability and reliability challenges that could be addressed with the following enhancements.

Better scalability with data compression. *SpamTracker*'s clustering algorithm is centralized, which raises scalability concerns, both for bandwidth (to exchange information between domains) and in terms of processing power (clustering complexity increases with input size). We are investigating ways to reduce load by distributing the clustering process. For example, compressing cluster information into average rows before sending this information to a centralized server may reduce bandwidth consumption: *SpamTracker* requires the full $IP \times domain$ matrix from each source to perform clustering, but requires only the average row vectors for each cluster (*i.e.*, the output of the algorithm) for classification.

Better reliability with replication and anycast. To improve availability, *SpamTracker* servers could be replicated and placed in different locations or on independent networks. Multiple servers might be anycasted or managed by different organizations (much like the DNS root nameserver infrastructure today), all of which perform the same computation and disseminate average rows to second-level servers, which in turn respond to user lookups.

6.4 Evasion

SpamTracker must be resistant to attacks that mislead the clustering engine in ways that can cause spam to be misclassified as legitimate email, and vice versa. To improve classification robustness, *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 over a secure channel. Although *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.

If spamming bots in a botnet select target domains from the *same distribution*, *SpamTracker*'s clustering algorithm will include these spammers in the same cluster. Still, *SpamTracker* is limited by the time window used for clustering (*e.g.*, 6 hours, as in Section 5), and a spammer might exploit this weakness to evade *SpamTracker*. We are improving *SpamTracker* to automatically adjust the window in response to the fraction of received email in the last window that was classified as spam. The intuition is that the fraction of spam does not change much over short timeframes, and a decrease in the fraction of flagged email indicates that the window is too small to cluster similar IPs together. Spamming bots might also try to emulate the distribution of target domains (or other behavioral features) of normal senders, but by doing so, they may be inherently less effective (*e.g.*, they may have to reduce their sending rate or the expansiveness of their target list).

6.5 Sensor Placement

A set of domains that observes more even sending behavior across domains may be able to better distinguish spammers from legitimate senders. Recall from Section 2.1.2 that 90% of the spam we observe is received by only 84 of the 115 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), we expect that a more even distribution of sensors email would further improve the *SpamTracker* classifier. Many commercial spam filtering companies (*e.g.*, IronPort [16], Secure Computing [31]) may already have this data. Another option for sensors would be ubiquitous Web mail domains such as `hotmail.com`, `gmail.com`, etc.

7. RELATED WORK

In this section, we discuss several areas of related work: (1) Previous characterization studies, several of which offer statistics that help build the case for behavioral blacklisting; (2) Existing systems for spam filtering, many of which use distributed monitoring but incorporate different algorithms for classification; (3) Previous approaches for classifying email into legitimate email and spam.

Blacklisting and identity. *SpamTracker* relates to previous blacklisting proposals. Conventional blacklists constitute lists of IP addresses of likely spammers and are intended to help spam filters [15, 23, 35] 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 [34]). 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 [24, 36, 37, 39]. These blacklists essentially maintain lists of IP addresses and must be vigilantly maintained so as to not going out of date. Sender Policy Framework (SPF) attempts to prevent IP addresses from sending mail on behalf of a domain for which they are not authorized to send mail [42], and domain keys associate a responsible identity with each mail [3]. Although both frameworks make it more difficult for an arbitrary IP address to send mail, they do not allow a recipient to classify an email sender with an unknown reputation.

Collaborative filtering and whitelisting. *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 [10, 12, 20, 27, 28, 40]. 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.

Other systems collect information from distributed sets of users either to help filter spam or decrease the probability that legitimate mail is mistakenly filtered. IronPort [17] and Secure Computing [32] sell spam filtering appliances to domains which then pass information about both legitimate mail and spam back to a central processing engine that in turn improves the filters. The widespread deployment of these products and systems make them ideal candidates for the deployment of an algorithm like *SpamTracker*.

Characterization studies. Our recent characterization study of the network-level behavior of spammers observes spamming behavior from the perspective of a single spam "trap" domain [30]. In this study, we observed 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 [13]. Our characterization of spammers in Section 2 builds on these previous studies by observing email sending patterns *across* domains and time.

Content-independent blocking. Like *SpamTracker*, Clayton's "spamHINTS" project also aims to characterize and classify spam with analysis of network traffic patterns, rather than email contents [38]. Earlier work on "extrusion detection" involves instrumenting a mail server with a log processing program to detect senders of spam both at the local ISP [8] and in remote ISPs [9]. 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 counts email volumes to identify emails that are likely sent from bots [6]; it 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 emails contents, such as the URLs contained in the bodies of the emails [4, 22]. Li *et al.* focus on clustering spam senders to predict whether a known spammer will send spam in the future [22], 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 [4]. 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* complements previous proposals that have suggests throttling senders using schemes such as stamps, proof-of-work, etc. One prominent postage scheme is called "bankable postage", whereby senders obtain stamps or tokens from some authority and then attach these tokens to emails [2, 41]. Other techniques for throttling spam require the sender to issue some "proof of work", either in CPU [5] or memory [14], although these schemes have also been criticized because, in certain circumstances, they can prevent legitimate users from sending normal volumes of email [21].

8. 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 clusters email senders based on the set of domains that they target. *SpamTracker* uses these sending patterns of confirmed spammers to build "blacklist clusters", each of which has an average vector that represents a spamming fingerprint for that cluster. *SpamTracker* tracks sending patterns of other 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 115 independent domains shows that *SpamTracker* can complement existing blacklists: it can distinguish spam from legitimate mail 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. Our ongoing work involves gathering data from a wider set of domains, improving the behavioral classification algorithms (*e.g.*, by using other features of email senders), and deploying the system to allow us to evaluate it in practice.

Acknowledgments. This work was supported by an NSF CAREER award CNS-0633974, by NSF Cybertrust awards CNS-0721581 and CNS-0716278, by NSF grant CCF-0721503, and in part by the Georgia Tech Algorithms and Randomness Center (ARC) ThinkTank (`http://www.arc.gatech.edu/`). We thank Merrick Furst, Suresh Ramasubramanian, Criag Sprosts, and Muhammad Mukarram bin Tariq for comments and help.

REFERENCES

[1] Spamhaus delisting policy, 2007. `http://www.spamhaus.org/sbl/policy.html`.

[2] M. Abadi, A. Birrell, M. Burrow, F. Dabek, and T. Wobber. Bankable Postage for Network Services. In *Proc. Asian Computing Science Conference*, Dec. 2003.

[3] E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton, and M. Thomas. *DomainKeys Identified Mail (DKIM) Signatures*, May 2007. `http://www.ietf.org/rfc/rfc4871.txt`.

[4] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker. Spamscatter: Characterizing Internet Scam Hosting Infrastructure. In *Proc. 16th USENIX Security Symposium*, Boston, MA, Aug. 2007.

[5] A. Back. Hashcash. `http://www.cypherspace.org/adam/hashcash/`.

[6] A. Brodsky and D. Brodsky. A Distributed Content Independent Method for Spam Detection. In *First USENIX Workshop on Hot Topics in Understanding Botnets (HotBots)*, Cambridge, MA, Apr. 2007.

[7] D. Cheng, R. Kannan, S. Vempala, and G. Wang. A divide-and-merge methodology for clustering. *ACM Transactions on Database Systems*, 31(4):1499–1525, 2006.

[8] R. Clayton. Stopping Spam by Extrusion Detection. In *First Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, July 2004.

[9] R. Clayton. Stopping Outgoing Spam by Examining Incoming Server Logs. In *Second Conference on Email and Anti-Spam (CEAS)*, Stanford, CA, July 2005.

[10] Cloudmark Authority Anti-Spam. `http://www.cloudmark.com/serviceproviders/authority/spam/`, 2007.

[11] Commtouch Inc. 2006 Spam Trends Report: Year of the Zombies. `http://www.commtouch.com/documents/Commtouch_2006_Spam_Trends_Year_of_the_Zombies.pdf`.

[12] E. Damiani, S. de Vimercati, and P. Samarati. P2P-Based Collaborative Spam Detection and Filtering. In *4th IEEE Conference on P2P*, 2004.

[13] Z. Duan, K. Gopalan, and X. Yuan. Behavioral Characteristics of Spammers and Their Network Reachability Properties. In *Proc. IEEE ICC*, Glasgow, Scotland, June 2007.

[14] C. Dwork and M. Naor. Pricing via Processing or Combatting Junk Mail. In *CRYPTO*, Santa Barbara, CA, Aug. 1992.

[15] P. Graham. Better Bayesian Filtering. `http://www.paulgraham.com/better.html`.

[16] IronPort. `http://www.ironport.com/`, 2007.

[17] IronPort Carrier Grade Email Security Appliance. `http://www.ironport.com/products/ironport_x1000.html`, 2007.

[18] R. Kannan, S. Vempala, and A. Vetta. On clusterings: good, bad and spectral. *J. of the ACM*, 51(3):497–515, 2004.

[19] Kelly Jackson Higgins, Dark Reading. Botnets Battle Over Turf. `http://www.darkreading.com/document.asp?doc_id=122116`, Apr. 2007.

[20] J. Kong et al. Scalable and Reliabile Collaborative Spam Filters: Harnessing the Global Socail Email Networks. In *3rd Annual Workshop on the Weblogging Ecosystem*, 2006.

[21] B. Laurie and R. Clayton. Proof-of-Work Proves Not to Work. In *Third Annual Workshop on Economics and Information Security (WEIS)*, Minneapolis, MN, May 2004.

[22] F. Li and M.-H. Hseih. An Empirical Study of Clustering Behavior of Spammers and Group-based Anti-Spam Strategies. In *3rd Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, July 2006.

[23] MailAvenger, 2007. `http://www.mailavenger.org/`.

[24] Mail Abuse Prevention System (MAPS). `http://www.mail-abuse.com/`.

[25] Messaging Anti-Abuse Working Group. MAAWG Issues First Global Email Spam Report. `http://www.prnewswire.com/cgi-bin/stories.pl?ACCT=104&STORY=/www/story/03-08-2006/0004316196`, Mar. 2006.

[26] PandaLabs Blog. Zunker Spamming Bot Front-end Analysis. `http://blogs.pandasoftware.com/blogs/pandalabs/archive/2007/05/08/Zunker.aspx`, May 2007.

[27] V. Prakash. Vipul's Razor. `http://razor.sourceforge.net/`, 2007.

[28] Pyzor. `http://pyzor.sourceforge.net/`, 2007.

[29] A. Ramachandran, D. Dagon, and N. Feamster. Can DNSBLs Keep Up with Bots? In *3rd Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, July 2006.

[30] A. Ramachandran and N. Feamster. Understanding the Network-Level Behavior of Spammers. In *Proc. ACM SIGCOMM*, Pisa, Italy, Aug. 2006.

[31] Secure Computing. `http://www.securecomputing.com/`, 2007.

[32] Secure Computing IronMail. `http://www.securecomputing.com/index.cfm?skey=1612`, 2007.

[33] A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. Information and Computation, 82:93–133, 1989.

[34] Spam and Open-Relay Blocking System (SORBS). `http://www.sorbs.net/`.

[35] SpamAssassin, 2007. `http://www.spamassassin.org/`.

[36] SpamCop. `http://www.spamcop.net/`.

[37] Spamhaus, 2007. `http://www.spamhaus.org/`.

[38] SpamHINTS. `http://www.spamhints.org/`, 2007.

[39] Realtime uri blacklist. `http://www.uribl.com/`.

[40] P. Vixie. Distributed Checksum Clearinghouse. `http://www.rhyolite.com/anti-spam/dcc/`, 2007.

[41] M. Walfish, J. Zamfirescu, H. Balakrishnan, and D. Karger. Distributed quota enforcement for spam control. In *Proc. 3rd Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, May 2006.

[42] M. Wong and W. Schlitt. *Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail*, Apr. 2006. RFC 4408.