SNARE: Spatio-temporal Network-level Automatic Reputation Engine

Nadeem Ahmed Syed
College of Computing,
Georgia Tech
nadeem@cc.gatech.edu

Nick Feamster
College of Computing,
Georgia Tech
feamster@cc.gatech.edu

Alexander G. Gray
College of Computing,
Georgia Tech
agray@cc.gatech.edu

Sven Krasser
Secure Computing
Corporation
skrasser@securecomputing.com

ABSTRACT

Current spam filtering techniques classify email based on content and IP reputation blacklists or whitelists. Unfortunately, spammers can alter spam content to evade content-based filters, and spammers continually change the IP addresses from which they send spam. Previous work has suggested that filters based on network-level behavior might be more efficient and robust, by making decisions based on *how* messages are sent, as opposed to what is being sent or who is sending them.

This paper presents a technique to identify spammers based on features that exploit the network-level spatio-temporal behavior of email senders to differentiate the spamming IPs from legitimate senders. Our behavioral classifier has two benefits: (1) it is *early* (i.e., it can automatically detect spam without seeing a large amount of email from a sending IP address—sometimes even upon seeing only a single packet); (2) it is *evasion-resistant* (i.e., it is based on spatial and temporal features that are difficult for a sender to change). We build classifiers based on these features using two different machine learning methods, support vector machine and decision trees, and we study the efficacy of these classifiers using labeled data from a deployed commercial spam-filtering system. Surprisingly, using only features from a single IP packet header (i.e., without looking at packet contents), our classifier can identify spammers with about 93% accuracy and a reasonably low false-positive rate (about 7%). After looking at a single message spammer identification accuracy improves to more than 94% with a false rate of just over 5%. These suggest an effective sender reputation mechanism.

1. INTRODUCTION

Current spam filtering systems commonly employ two complementary mechanisms: content filters and sender reputation. Although content filters are effective at blocking certain classes of unwanted email messages, they are also quite brittle. For example, spammers can evade content filters by embedding messages in images or other file formats (e.g., PDF, MP3); as a result, network administrators may have an incredibly difficult time maintaining content filters. Accordingly, spam filters also rely on sender reputation to filter messages: in other words, they use the reputation of the sender to help determine whether the message being sent is legitimate or spam, sometimes even as a means for filtering email before it is even accepted for delivery.

A common approach to IP-based filtering is the use of DNS blacklists (DNSBLs), e.g. SpamHaus [6]. DNSBLs maintain lists of IP addresses that are known to send spam, which is based on information retrieved from spam traps and manual listings by humans. Unfortunately, DNSBLs have several shortcomings. First, the reaction time for IP addresses being listed on an DNSBL is slow. This is due to the fact that a spammer either needs to send a message to a spam trap used by an DNSBL or a human needs to manually list a spamming IP address. Therefore, there may be significant delay between the time when a spammer becomes active and the time when that spammer is ultimately listed in an IP blacklist. Second, the IP addresses of email senders is continually changing due to dynamic addressing, newly compromised hosts that might be used to send spam, etc. Thus, due to the slow reaction time, DNSBLs often times are not able to react on fast enough. Although content-based spam identification methods have been extensively studied [15, 16] and are quite mature, methods based on network-level behavior of spammers are relatively less developed.

This paper presents *SNARE* (Spatio-temporal Network-level Automatic Reputation Engine), a sender reputation system that can quickly, accurately, and automatically classify email senders based on features that can be determined early in a sender's history—sometimes even upon seeing only a single packet. The key insight underlying *SNARE*'s reputation mechanism is to classify senders based on *how* they are sending messages (i.e., traffic patterns), rather than *who* the senders are (i.e., their IP addresses). Such features are typically more invariant, because it is more difficult for a spammer to change the mechanism for sending messages than it is to send messages from a fresh set of IP addresses; by focusing on such features, *SNARE* can potentially be more evasion-resistant than existing sender reputation mechanisms.

Despite the appeal of relying on invariant "network-level" features that can be used to build an automated classifier, the major challenge of identifying *which* features most efficiently and effectively distinguish spammers from legitimate senders remains. In particular, ideally our system should have the following properties:

- *Accurate.* It should accurately distinguish spammers from legitimate senders.
- *Early and adaptive.* It should be able to classify senders based on minimal historical information.
- *Evasion-resistant.* It should be based on features that are difficult for a sender to evade.

This paper evaluates eleven different network-level features that require varying levels of information about senders' previous history.

We evaluate *SNARE* using information about email senders collected from Secure Computing's TrustedSource, a commercial reputation system that filters spam for more than 8,000 distinct domains. We categorize *SNARE*'s features according to four categories, in increasing order of efficiency: (1) those that rely only on historical activity from other senders but not the sender in question; (2) those that require limited historical information from the sender; (3) those that require at least 24 hours of historical information about the sender in question. Surprisingly, using only features from a single IP packet header, *SNARE* can identify spammers with more than 90% accuracy with a reasonably false-positive rate for sender reputation (7%). The prediction accuracy on spammers improves to more than 94% with a a low false-positive rate of about 5% with features extracted from a single message. Additionally, our analysis of the rules extracted from the decision tree classifier shows that the combination of *geodesic distance* between the sender and the recipient and the *email server density* around the sender is very effective in separating out a lot spam senders from legitimate mail servers. Our results also suggest that *SNARE* might ultimately be deployed as a sender reputation system in conjunction with other tools (e.g. content-based filters) and used as a first-pass filter.

The rest of this paper is organized as follows. Section 2 presents background on existing sender reputation systems and a possible scenario where *SNARE*'s algorithms could be deployed. Section 3 presents an overview of the network level behavioral properties of spammers and legitimate senders used in *SNARE*'s classification algorithms and presents first-order statistics concerning these features. Section 4 describes how we incorporate these behavioral features into two different classifiers: a decision tree, and a support vector machine. Section 5 evaluates *SNARE*'s performance using different feature subsets, ranging from those that can be determined from a single packet to those that require varying amounts of history. Section 8 describes related work, and Section 9 concludes.

2. BACKGROUND

In this section, we provide background on existing sender reputation mechanisms and present motivation for improved sender reputation mechanisms. We also describe Secure Computing's TrustedSource system, which is both the source of the data used for our analysis and a possible deployment scenario for *SNARE*'s classification algorithms.

2.1 Blacklists and Email Reputation Systems

Today's email spam filters traditionally perform lookups to DNS-based IP blacklists (DNSBLs) to determine whether a specific IP address is a known source of spam at the time of lookup). One commonly used public blacklists is Spamhaus [6], which maintains three main blacklists: the SBL is a list of known spam sources, the XBL is a list that enumerates hosts believed to be compromised with third-party exploits (such machines can often serve as open proxies and have been known to send spam), and the PBL is a "policy block list", which enumerates IP addresses that are typically not allowed to make connections to mail servers (e.g., dynamic IP address ranges). Other blacklist operators include SpamCop [5] and SORBS [4].

These blacklists are typically called DNS-based blacklists because they provide a DNS interface for querying specific IP addresses. A spam filter might issue a DNS query for an IP address `10.0.0.1` by sending an A-record query for `1.0.0.10.sbl.spamhaus.org`. If the IP address is not blacklisted, the query will return an `NXDOMAIN`. If, on the other hand, the IP address is listed, the DNSBL will return an IP address, where different IP addresses reflect statuses.

Current blacklists have many shortcomings. First, they only provide information based on IP addresses. Unfortunately, as an earlier work has observed [23], the IP addresses of senders are dynamic: roughly 10% of spam senders on any given day have not been previously observed. This study also observed that many spamming IP addresses will go inactive for several weeks, presumably until they are removed from IP blacklists. This level of dynamism makes maintaining responsive IP blacklists incredibly difficult; they are also often coarse-grained, blacklisting entire prefixes—sometimes too aggressively—rather than individual senders. One goal of *SNARE* is to develop a reputation system that is both automated and faster than existing blacklists. Second, IP blacklists are typically incomplete. A previous study has noted that as much as 20% of spam received at spam traps is not listed in any blacklists [22]. Finally, anecdotal evidence is rife with stories of IP addresses of legitimate mail servers

Field	*Description*
timestamp	UNIX timestamp
ts_server_name	Name of server that handles the query
ip_score	Score for the queried IP
score	Score for the message based on a combination of IP score and a content-based score
source_ip	Source IP in the packet (DNS server relaying the query to us)
device_serial	Serial number of the device that sent the query
query_ip	The IP being queried
body_length	Length of message body
count_taddr	Number of To-addresses

Table 1: Description of data used from the Secure Computing dataset.

being incorrectly blacklisted (e.g., because they were reflecting spam to mailing lists). At this point, several commercial reputation systems exist that take additional data such as SMTP meta data or message fingerprints into account to address these shortcomings [7].

Previous work introduced the idea of "behavioral blacklisting" and developed a spam classifier based on a single behavioral feature: the number of messages that a particular IP address sends to each recipient domain [23]. Although we do incorporate this feature into some version of *SNARE*, we note that this feature has several shortcomings. First, it is not as responsive as other features, because it requires each candidate sender to send messages to several recipient domains before it can be accurately classified. Second, it is not particularly evasion-resistant: spammers could potentially randomize (or continually change) the recipient domains that each sending IP address targets.

This paper builds on the main theme of behavioral blacklisting by finding better features that can classify senders earlier and are more resistant to evasion.

2.2 Secure Computing TrustedSource

This section describes Secure Computing's TrustedSource reputation system. We describe how we use the data from this system to evaluate *SNARE*'s classification algorithms. We also describe how *SNARE*'s algorithms might be incorporated into a real-time sender reputation system such as TrustedSource.

TrustedSource is a commerical reputation system allowing lookups on various Internet identifiers such as IP addresses, URLs, domains, or message fingerprints. It gets query feedback from various different device types such as mail gateways, web gateways, and firewalls.

2.2.1 Data

We evaluated *SNARE* using a subset of two full days of data from the query logs from Secure Computing's TrustedSource system. Table 1 summarizes the fields that we use to develop and evaluate *SNARE*'s classification algorithms. Each received email generates a lookup to the TrustedSource database, so each entry in the query log essentially represents information about a received email message. The `timestamp` field reflects the time at which the message was received at a TrustedSource appliance in some domain; the `source_ip` field reflects the source IP of the machine that issued the DNS query (i.e., the receiving appliance); `device_serial` reflects the identity of the spam filtering appliance that issued the query. We use `device_serial` as a substitute for the recipient domain, since we do not have a field that explicitly represents the receiving domain. The `query_ip` field is the IP address being queried (i.e., the IP address of the email sender). We use many of the other features in Table 1 as input to *SNARE*'s classification algorithms. Due to the sheer volume of the full set of logs, we used data from one specific TrustedSource server logs, as opposed to using the data from all the servers.

The data contains a `ip_score` field that indicates how TrustedSource ultimately scored the sender based on its current reputation system (essentially, either spam, legitimate email, or uncertain). We use the `ip_score` as ground truth labels to train *SNARE*. Our goal is to develop a fully automated classifier that is as accurate as TrustedSource but relies only on early, evasion-resistant network-level features that can be extracted at low cost on commodity hardware.

2.2.2 Deployment scenario

Because it operates only on network-level features of email messages (i.e., it does not use email contents for either training or classification), *SNARE* could be deployed either as part of TrustedSource or as a standalone DNSBL. As with TrustedSource, however, email servers that issued lookups to *SNARE* would have to issue modified DNS queries that contained additional meta-data about the received messages.

3. NETWORK-LEVEL BEHAVIORAL PATTERNS

In this section, we explore some important spatio-temporal properties of email senders and discuss why these properties are relevant and useful for diferentiation between spammers and legitimate senders. We also consider how these behaviors can yield different kinds of features depending on the amount of history available from any specific sender. Table 2 summarizes all the features that can be extracted using only non-content information about the messages and with varying amount of available historical data. We describe these features in terms of increasing overhead. Section 3.1 describes features that can be determined with no previous history from the sender the SNARE is trying to classify, and given only a *single packet* from the IP address in question; Section 3.2 describes features that can be gleaned from a *single SMTP message header*; Section 3.3 features that can be derived with varying amounts of history (i.e., aggregates of other features). Note that no particular feature needs to be perfectly discriminative between ham and spam on its own: SNARE's classification algorithm (described in Section 4) uses a combination of these features to build the best classifier. We evaluate classifiers using different sets of these features in Section 5.

3.1 Single Packet

In this section, we discuss some properties that can help identify a spammer even when we have not seen the particular IP address (we will call it *candidate IP*) before, but some history of network-level activities of other email servers is available. We first discuss the features that can be extracted from just a single IP packet: the geodesic distance between the sender and receiver, sender neighborhood density and probability ratio of spam to ham at the time the IP packet arrives. Next we discuss other features that can be extracted from a single message: the number of recipients in the message, and the length of the message.

3.1.1 Sender-receiver geodesic distance

Recent results [8, 14] suggest that social networking structure among communicating parties could be utilized to effectively isolate spammers. The challenge with this approach is to find features that can be calculated quickly on the wire and deployed in a practical system. To this end, we hypothesize that the legitimate emails tend to travel shorter distances in geographic terms, whereas the distance traveled by spam will be random. In other words, a spam message may be just as likely to travel a short distance as across the world.

Figure 1(a) shows the distribution of the distance between the sender and the target IP addresses for each of the five categories of messages. The distance used in these plots is the geodesic distance, that is, the distance along the surface of the earth. It is computed by first finding the physical latitude and longitude of the source and target IP using the MaxMind's GeoIP database [18] and then computing the distance between these two points. These distance calculations assume that the earth is a perfect sphere. For *certain ham* 90% of the messages travel about 2200 miles or less. On the other hand for *certain spam*, only 25% of messages stay within this range. In fact if we look at the 90% level, the messages travel more than 6000 miles, which is a quarter of the earth's circumference at the equator. Based on these plots, geodesic distance certainly looks like a feature that can be used to distinguish spam from ham and can be computed quickly using just a single IP packet.

3.1.2 Sender neighborhood density

A large percentage of spam messages today is generated by botnets [22, 26]. For messages originating from the same botnet, it can be expected that the infected IP addresses all lie close to each other in the numerical space, very likely within the same subnet. One possible way of detecting whether an IP belongs to a botnet is to look at the past history and determine if messages have been received from other IPs in the the same /xx subnet as the current sender, where xx can be determined experimentally. If we find many different IPs from the same subnet, the chances that the whole subnet is infested with bots are very high. The problem with this approach is that the frame of reference is limited to a subnet and the lookup does not transcend the subnet boundaries.

A better measure of *email server density* in an IP's neighborhood is the distances to its k nearest neighbors. The distance to the k nearest neighbors can be computed by treating the IPs as set of numbers from 0 to $2^{32} - 1$ (for IPv4) and finding the nearest neighbors in this single dimensional space. We can expect these distances to exhibit different patterns for spam and ham. If the neighborhood is *crowded* these neighbor distances will be small indicating possible presence of botnet. On the other hand if the IP is not part of local botnet, then we shouldn't see too many IPs within short distances. Another benefit of this measure is that it also pulls out IP addresses that are operating in isolated parts of the IP space.

Figure 1(b) shows the average distance to the first twenty nearest neighbors in the historical log for each category of senders. The plots reflect the fact that a large majority of spam originates from botnets which have high email server density in a given IP region. The distance to the k^{th} nearest neighbor for spam tends to be much shorter on average than it is for legitimate senders. This is the opposite of geodesic distance measure where the distances for the legitimate senders tend to be smaller than those for spammers. The plots also confirm the intuitive sense that this is a feature that can help identify spamming botnets and therefore distinguish spamming IPs from legitimate senders quite well.

3.1.3 Time of day

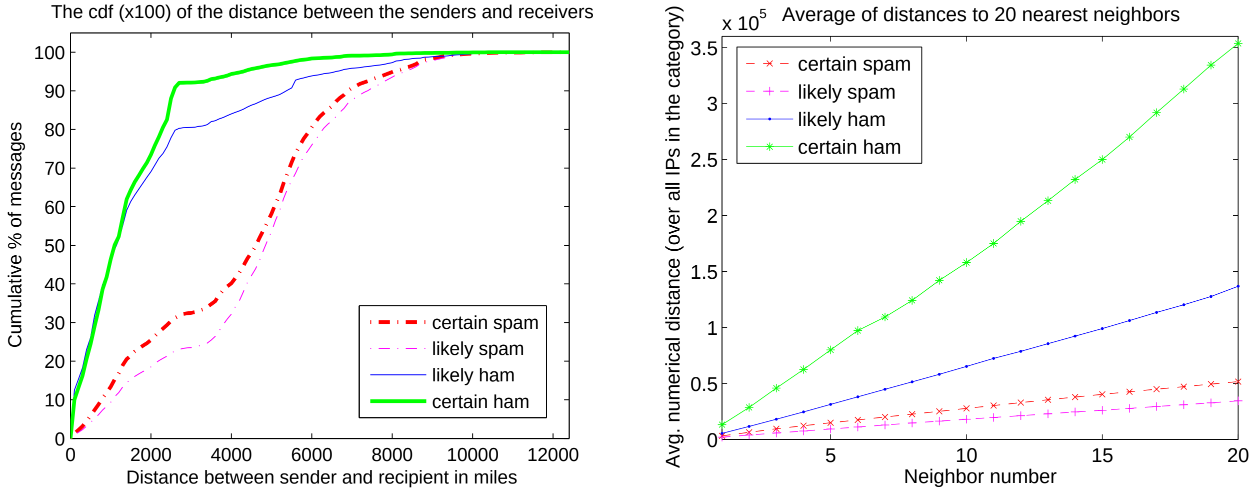
Another feature that can be extracted using information from single IP packet is the message timestamp. The time of message arrival could potentially bias the decision towards either spam or ham. Can the timestamp be used to influence this decision? To make this analysis accurate, the timestamp for each message has been corrected to correspond to the time at the sender's physical location, as opposed to Coordinated Universal Time (UTC) used in the log.

Figure 2(a) shows the relative percentage of messages of each type at different times of the day. Here again the legitimate senders and the spam senders show different diurnal patterns. Two times of the day are particularly striking: the relative amount of ham tends to ramp up quickly at the start of the work-day and peaks early morning. It goes down relatively quickly as well at the end of the work day. On the other hand spam goes up at a slower steady pace, probably as machines are switched on in the morning. The increase in relative volume for spam could also be later if the bots use a lot of home machines that switch on later. The spam volume stays steady through the day and starts dropping around 9:00 pm, probably when machines are switched off again.

To utilize the timestamp as a feature, the probability ratio of spam to ham, at the time of the day when the message is received is used. This ratio is calculated for each hour of the day as follows:

	Single Message Features			Aggregated Features	
Feature (*Section in text*)	**Single-Packet**	**Header**	**Message**	**2+ Messages**	**1+ Days**
Features that Do Not Require History					
Geodesic distance between the sender and recipient (*3.1.1*)	Any	Any	Any	—	—
Number of recipients (*3.1.4*)	—	Any	Any	—	—
Message length in bytes (*3.1.5*)	—	—	Any	—	—
Features that Require History only from Other Senders					
k nearest neighbors of the IP (in numerical space) (*3.1.2*)	1+ Hour	1+ Hour	1+ Hour	1+ Hour	1+ Hour
probability ratio of spam to ham at the time of message (*3.1.3*)	1+ Day	1+ Day	1+ Day	—	—
Features that Require History from the Candidate Sender					
Mean and variance of geodesic distance (*3.2.1*)	—	—	—	Any	Any
Mean and variance of number of recipient (*3.2.1*)	—	—	—	Any	Any
Mean and variance of message size (in bytes) (*3.2.1*)	—	—	—	Any	Any
# messages sent to 100 most frequent target domains (*3.2.2*)	—	—	—	1+ Hour	1+ Hour
Time-series of *number of messages* in fixed-size bins (*3.2.3*)	—	—	—	—	Any
Time-series of *number of bytes* in fixed-size bins (*3.2.3*)	—	—	—	—	Any

Table 2: The set of all the features used in *SNARE*.



(a) Geodesic distance between the sender and recipient's geographic location

(b) Average of numerical distances to the 20 nearest neighbors in the IP space

Figure 1: Spatial differences between spammers and legitimate senders

spam probability (or ham probability) =
# of spam (or ham) messages received in the given hour of the day / Total number of spam (or ham) messages received over the whole day(s).

The probability ratio is then simply

spam probability / ham probability .

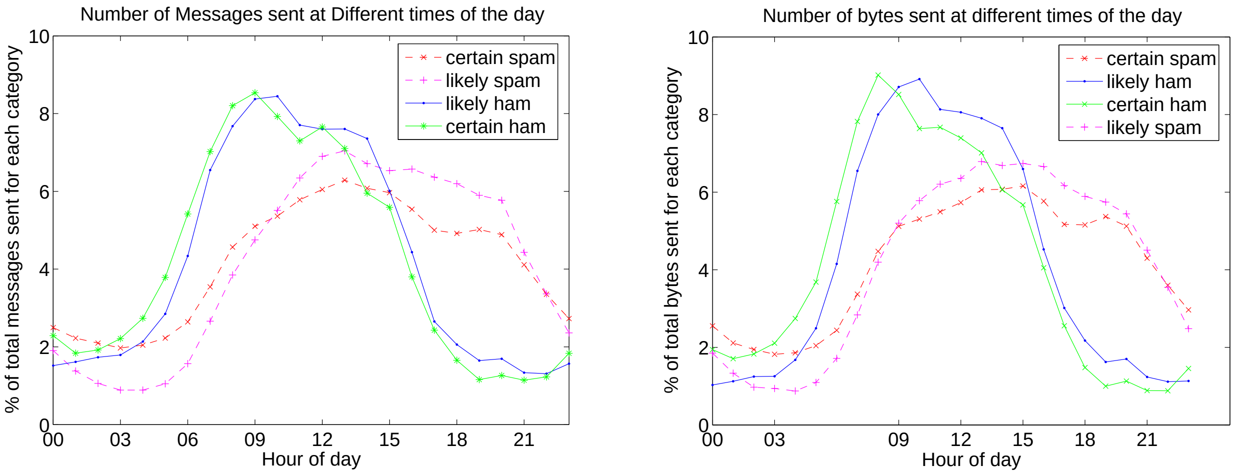
When a new message is received, the precomputed spam to ham probability ratio for the corresponding hour of the day can be looked up and used as a feature. This ratio can be recomputed on a daily basis.

3.2 Single Header and Single Message

Once connection request gets accepted, and the SMTP header and subsequently, the compete message are received, additional non-content features can be extracted. We discuss two of these possible features that could be used for spammer identification below.

3.2.1 *Number of recipients*

The features discussed so far can be extracted from a single IP packet from any given specific IP address combined with some historical knowledge of messages from other IPs. Another feature available without looking into the content is the number of address in "To" field of the header. This is a feature that can be extracted after receiving the whole SMTP header but before accepting the message body. Figure 3 shows the distribution of number of addresses on the "To" field for each category of messages. Both the axes are on log-scale to focus the plot on the smaller values where most of the data is concentrated. Based on this plot and looking at the actual values, it appears that if there are very large number of recipients on the "To" field (100 or more), then the message is almost certainly not spam. Beyond that, there doesn't seem to be a significant difference between the different types of senders for this measure. The noticeable



(a) Number of messages sent at different times of the day

(b) Number of bytes transmitted at different times of the day

Figure 2: Differences in diurnal messaging patterns of spammers and legitimate senders

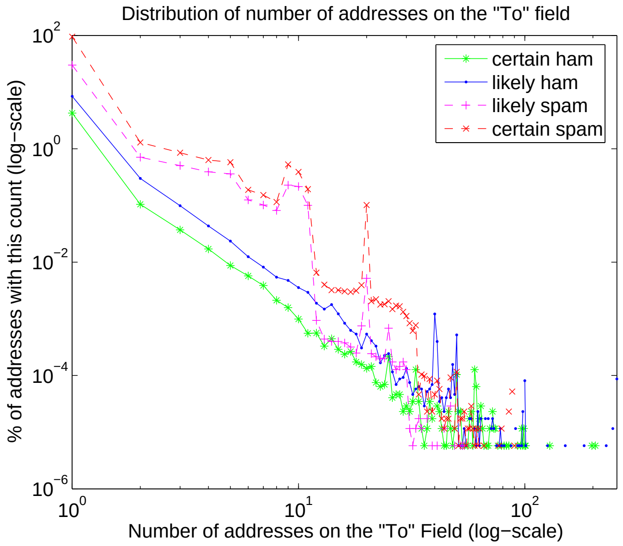


Figure 3: Distribution of number of addresses listed on the "To" field for each category

differences around 4 to 12 addresses are in fact very small, given that this is log-scale. So this feature is not expected to be useful in telling spammers apart and we will validate this experimentally in the next section.

3.2.2 Message length

Once an entire message has been received, the email body size in bytes is also known. Since a given spam sender will mostly send same or similar content in all the messages, it can be expected that the variance in the size of messages sent by a spammer will be lower than among the messages sent by a legitimate sender. To stay effective, the spam bots also need to keep the message size small so that they can maximize the number of messages they can send out. As such the spam messages can be expected to be biased towards the smaller size. Figure 4 shows the distribution of messages for each category. The spam messages are all clustered in the 1Kb-10Kb range whereas the distribution of message size for good senders is more evenly distributed. Consequently, the message body size is another property of messages than can help differentiate between ham and spam.

3.3 Aggregate Features

The behavioral properties discussed so far can all be constructed using a single message. If some of history from an IP is available, some *aggregate IP level features* can also be constructed. We now discuss these in the following subsections.

3.3.1 Mean and variance of single-message features

With availability of information about multiple messages from a single IP, the overall *distribution* of the following measures can be captured by using a combination of mean and variance:

1. geodesic distance between the sender and recipient
2. number of addresses in the "To" field of the SMTP header
3. message body length in bytes

By virtue of summarizing behavior over multiple messages and over a time interval, these aggregate features will likely yield a more reliable IP reputation prediction. On the flip side, these come at the cost of increased latency as we need to collect a number of messages before we compute these. By averaging over multiple messages they may also smoothen out the structure of the feature space and possibly make the marginal cases harder to classify.

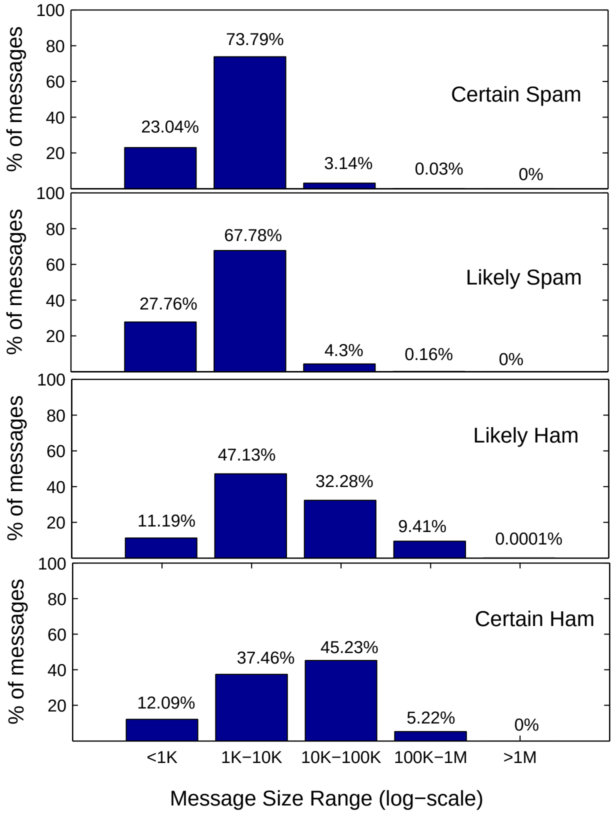


Figure 4: Histograms of message sizes for the different categories of messages

3.3.2 *Distribution across recipient domains*

Another property of messaging IPs that can be extracted at an aggregate level is the number of messages sent to a set of target domains by a particular IP. Ramachandran *et al.* [23] used this feature in the design of the SpamTracker system, based on their earlier finding [22] that many spamming senders distribute spam across multiple domains to keep the volumes at each individual domain low and hence evade detection. So, monitoring the number of email volumes from each sender across a set of target domains should be useful in identifying these kind of spammers. To construct this feature we identify all the target domains that had received messages over the two days of data used for evaluation, then picked the top 100 domains that had received most number of emails. Finally, for each IP we counted the number of messages sent to each domain over a twenty-four hour duration.

3.3.3 *Timeseries of message volume*

As Figure 2 shows, the spam senders clearly have a more smoothly varying message volumes over the day when compared to the legitimate senders. If we have accumulated log of messages from an IP for 24 hours or more, these can be used to construct a time-series of:

1. number of emails sent in each time-interval.
2. number of bytes sent in each interval.

When constructing a time-series, one challenge is the choice of appropriate time interval for binning the data. Manually selected bin widths may not give the best performance—statistical methods that minimize some relevant loss function to obtain the optimal bin size may be more appropriate. We investigated a promising automated binning method, autoPSTH [24]), to determine the optimal bin-width for each type of timeseries.

The autoPSTH determined optimal bin-width of 39 minutes was compared with three other manually selected bin sizes of 30, 60, and 180 minutes. Timeseries were constructed using each bin-width and single day of data and used to train a classifier. The prediction accuracy of the trained classifiers was evaluated to determine the most suitable bin-width. Ultimately, the best performance was obtained using 3 hour bins. This is the bin-size we use in this paper. Apparently, autoPSTH's objective function does not yield bin widths that are also optimal for classification.

4. MACHINE LEARNING METHODS

With a better understanding of the behavioral differences between the spam and ham at the network level , we can design an IP reputation system that utilizes these differences. Since we have labeled data, a supervised machine learning method to classify spam and ham is most appropriate. We evaluate and compare spam filters built using two such classifiers, Support Vector Machine (SVM) [10] and Decision Tree [20], each chosen for its different strengths in relation to this problem. Though there are many other classifiers that could have been applied, these represent the two most interesting points in the space of classifiers; SVM is most accurate, decision tree is most interpretable. Before discussing the rationale behind the choice of these two classifiers, we provide a brief introduction to these below.

4.1 Support Vector Machine

For a binary classification problem, a Support Vector Machine (SVM) algorithm constructs a maximum margin hyperplane to separate the data points of the two classes. Given n data points and their class labels $\{(\mathbf{x_1}, y_1),...,(\mathbf{x_n}, y_n)\}$, where each $\mathbf{x_i}$ is a vector of attributes or features, SVM training involves solving a quadratic programming problem and the optimal solution gives rise to a decision function of the following form:

$$f(\mathbf{x}) = sign\left[\sum_{i=1}^{n} y_i\alpha_i(\mathbf{x}\cdot\mathbf{x_i}) + b\right].$$

When the given data is not separable by a hyper-plane, one can first non-linearly transform the set of input training vectors $\mathbf{x_1}, ..., \mathbf{x_n}$ into a higher dimensional feature space using a map $\mathbf{\Phi}(\mathbf{x_i}) \mapsto \mathbf{z_i}$ and then do a linear separation in that space. This leads to a decision function of the form:

$$f(\mathbf{x}) = sign\left[\sum_{i=1}^{n} y_i\alpha_i(\mathbf{\Phi}(\mathbf{x})\cdot\mathbf{\Phi}(\mathbf{x_i})) + b\right]$$

$$= sign\left[\sum_{i=1}^{n} y_i\alpha_i\mathbf{K}(\mathbf{x}\cdot\mathbf{x_i}) + b\right]$$

where $\mathbf{K}(\mathbf{x}.\mathbf{x_i}) = \mathbf{\Phi}(\mathbf{x}).\mathbf{\Phi}(\mathbf{x_i})$ is known as a *kernel*. Notice that training and classification procedures now involve computing high dimensional dot products of the form $\mathbf{\Phi}(\mathbf{x}).\mathbf{\Phi}(\mathbf{x_i})$. To avoid these computations we consider only those classes of $\mathbf{\Phi}$ that reduce to simple kernel function $\mathbf{K}$.

Often, only a small fraction of the α_i coefficients are nonzero. The corresponding pairs of $\mathbf{x_i}$ entries (known as *support vectors*), y_i output labels, α_i coefficients, and offset b fully define the decision function and are preserved for use in the classification procedure.

For this study we have chosen to use the *Gaussian kernel*:

$$\mathbf{K}(\mathbf{x},\mathbf{x}') = exp\left(-\frac{||\mathbf{x}-\mathbf{x}'||^2}{2\sigma^2}\right),$$

where σ is a parameter that needs to be tuned.We use Gaussian kernel because it is widely used in the machine learning literature as the default kernel, has demonstrated excellent performance on a range or classification problems and involves only a single tweak parameter, σ. The SVM^{light} [17] implementation is used for training the SVM classifier.

4.2 Decision Tree

Decision tree algorithm builds a decision tree from a set of labeled training data $\{(\mathbf{x_1}, y_1),...,(\mathbf{x_n}, y_n)\}$. As before, each data point $\mathbf{x_i}$ is a vector of attributes. Decision tree algorithm constructs a classification tree by successively splitting the data into smaller subsets along a selected attribute. In particular, C4.5 [21], the decision tree algorithm we use in this study, uses the *information gain* criterion to choose an attribute for splitting the data:

$$IG(S_i) = -\sum_{j=1}^{k}\frac{freq(C_j,S_i)}{|S_i|}\times\log_2\left(\frac{freq(C_j,S_i)}{|S_i|}\right),$$

where k is the number of target classes (2 in our case), S_i is the subset of data points at node i, $|S_i|$ is number of points in S_i and $freq(C_j,S_i)$ is the number of points of class j in the set S_i. At any given node, the attribute with the highest normalized information gain is the one used to make decision at that node. Each of the leaf nodes correspond to one of class labels. Classifying a new data point involves simply traversing a path through the tree based on the values of the features and predicting the class as the label of the leaf node on the traversed path.

4.3 Choice of Learning Algorithm

Support vector machines choose themselves for any classification problem since they have been the standard bearers for prediction accuracy over the last decade. They have been shown empirically to give good generalization performance on a wide variety of problems such as handwriting recognition, face detection, text categorization, etc. On the other hand, they do require significant parameter tuning before the best performance can be obtained. An appropriate kernel needs to chosen and the best parameters to for that kernel need to be determined experimentally. The worst case training time of SVM can be $O(N^3)$, where N is the number of training points. Additionally, if the training set is large and big percentage of α_i turn out to non-zero, the classifier itself can take up lot of storage space and the classifying new data points will be correspondingly slower since the classification cost is $O(S)$ for each test point, where S is the number of support vectors.

Decision tree algorithms have been popular classification method for almost two decades since they possess certain desirable properties; the resulting classifier is simple to understand and interpret, they are able to easily handle both continuous and discrete feature values, and the trained classifier scales well with the number of training samples, with the prediction on a new test point being $O(log(n))$, where n is the number of nodes in the trained tree. Decision tree is also faster to train with training cost of $O(NlogN)$. On the other hand, a decision tree algorithm may not beat a *well tuned* support vector machine in terms of prediction accuracy, especially when all the features are continuous (and not mixed or discrete). Given that our feature set is a mixture, with many features that are count values and hence discrete and others that are continuous, decision trees are overall preferable in this context.

We further discuss how these difference manifest themselves in our problem and their implications in our discussion section when we compare their performance on the reputation prediction problem.

5. RESULTS AND DISCUSSION

In this section we evaluate the performance of *SNARE* using different levels of information that result in a reputation prediction system with different level of responsiveness, right from identification using a single packet to more a more reactive reputation score using historical data.

Across the board the support vector machine had significantly better performance when the features were log-transformed before training the SVM. Hence, for all the experiments in this paper, the logarithm of the features is used to train SVMs. All the classifiers are evaluated using 10-fold cross-validation. This is done by splitting the dataset into 10 subsets, training on 9 subsets of the data and using the remaining subset as a test set. Different test set is used in each of the 10 rounds and the prediction accuracy and false positive rates are calculated as the average of the 10 folds. The term *false-positive rate* is used to indicate the percentage of legitimate senders that were classified as spammers.

	Confusion Matrices for SVM prediction			Confusion Matrices for decision tree prediction		
		Classified as			Classified as	
		Spam	Ham		Spam	Ham
6 neighbors	Spam	25983(86.61%)	4017(13.39%)	Spam	27315(91.05%)	2685(8.95%)
	Ham	3700(12.33%)	26300(87.67%)	Ham	2693(8.98%)	27307(91.02%)
		Classified as			Classified as	
		Spam	Ham		Spam	Ham
10 neighbors	Spam	26535(88.45%)	3465(11.55%)	Spam	27665(92.22%)	2335(7.78%)
	Ham	3118(10.39%)	26882(89.67%)	Ham	2318(7.77%)	27682(92.27%)
		Classified as			Classified as	
		Spam	Ham		Spam	Ham
20 neighbors	Spam	27561(**91.87%**)	2439(08.13%)	Spam	27820(**92.73%**)	2180(07.27%)
	Ham	2730(**9.10%**)	27270(90.90%)	Ham	2161(**7.20%**)	27839(92.80%)

Table 3: *SNARE* performance using single IP packet

5.1 Single Packet

When a mail server receives a new connection request, the server can provide *SNARE* with the IP addresses of the sender and the recipient and the time-stamp. Recall from Section 3 even if *SNARE* has never seen this IP before, it can still combine this information with recent history of behavior of other email servers and construct the following features:

- geodesic distance between the sender and the recipient,
- distances to k nearest neighbors of the sender in the log, and
- probability ratio of spam to ham at the time the connection is requested

5.1.1 Predictive performance of SVM & Decision Tree

To evaluate the effectiveness of these features, we trained both support vector machines and decision trees on these features. The number of nearest neighbors used to train the classifier were also varied to determine its effect on prediction accuracy. Table 3 shows the confusion matrices for the different configurations. The best spammer identification rate and false positive rate for each type of classifier is highlighted in bold.

The classifier built using a decision tree and trained using 20 nearest-neighbors has a surprisingly strong performance. It correctly identifies almost 93% of the spamming IPs while having a reasonably low false-positive rate. Just over 7% of legitimate IPs get labelled as spammers. This is a significant result since it uses features constructed from limited amount of data and just a single IP packet from the candidate IP. Sender reputation system will be deployed in conjunction with a combination of other techniques including content based filtering. As such, as a first line of defense, this system will be very effective in eliminating a lot of undesired senders. In fact once a sender is determined to be spammer, the mail server does not even need to accept the connection request, saving network bandwidth and computational resources.

Table 3 also shows that decision tree tends to perform better than support vector machines for this set of features. This could possibly be due to the fact that we have a mixture of discrete and continuous features, whereas the Gaussian kernel assumes all the features to be smoothly varying continuous features. Unfortunately, there is no default support vector machine kernel that can effectively handle mixed attributes well.

As for the appropriate choice of k of the nearest neighbor distances, while the decision tree and SVM predictions improve noticeably as we increase k from 6 to 10, the improvement is not that significant when k is increased from 10 to 20, especially for the decision tree. So, it is reasonable to conclude that increasing the number of neighbors beyond 20 will not get us a much improved performance, but it will increase the dimensionality of the training points and hence the complexity of the trained classifier. With increased dimensionality, correspondingly more training points will also be needed to train the classifier. Consequently, we use $k = 20$ for rest of this paper.

5.1.2 An additional benefit: Evasion-resistance

An important advantage of *SNARE* is that the combination of attributes is difficult for spammers to modify; hence, the resulting reputation mechanism is evasion-resistant, as we discuss below.

Geodesic distance: The distribution of geodesic distances between the spammers' physical location and their target IP's location is a result of the spammers' necessity to reach as many target mail boxes as possible and in the shortest possible time. If they had to modify their behavior so that their geodesic distances are biased towards their neighborhood, they will be limited to predominantly spamming domains that are hosted on servers located in a physical vicinity. This will severely limit the ability of botnets as they will need to infect machines in each region of the world that they need to send messages to.

Nearest neighbor distances: Nearest neighbor distances is another feature set that will be hard to modify. Distances

		Confusion Matrices for SVM prediction		Confusion Matrices for decision tree prediction	
		Classified as		Classified as	
		Spam	Ham	Spam	Ham
Header	Spam	27556(91.85%)	2444(8.15%)	27768(92.56%)	2232(7.44%)
	Ham	2715(9.05%)	27285(90.95%)	2171(7.24%)	27829(92.76%)
Message	Spam	28239(94.13%)	1761(5.87%)	28319(**94.40%**)	1681(5.60%)
	Ham	1842(6.14%)	28158(93.86%)	1572(**5.24%**)	28428(94.76%)

Table 4: *SNARE* performance using features from a single header and single message

to k nearest neighbors effectively isolate existence of unusually large number of email servers within a small sequence of IP addresses. If the spammers try to alter their neighborhood density, they will not be able to use too many machines within a compromised subnet. This will significantly reduce the IP space available to infect with bots and disrupt the ability of spam bots to proliferate.

Time of day message probability: This is one feature which appears to be less resistant to evasion tactics compared to the previous two. Having said that, spamming botnets' diurnal pattern is tied to when the infected machines are switched on. For botnets to modify their diurnal message volumes over the day to match the legitimate message patterns, they will have to lower their spam volume in the evenings, especially between 3:00 pm and 9:00 pm and also reduce email volumes in the afternoon. This will again reduce the effectiveness of the botnets to send large amounts of email.

Overall, in combination, these three behaviors are quite difficult to modify without significantly reducing the effectiveness of spam-bots. They are also easy to compute and don't have a high computational or storage requirements, making this version of *SNARE* straightforward to deploy in a practical sender reputation system.

5.2 Single Header and Single Message

Single packet features allow *SNARE* to rapidly identify and drop connections from spammers even before looking at the message header. Once a mail server has accepted the connection and examined at the whole message, we can make further determination of sender reputation with increased confidence. As described in Section 3.2, these features include the number of recipients and message body length.

Table 4 shows the prediction accuracy for each kind of classifier when we combine the features from the previous section with these additional features. As the results from Section 3 suggest, adding the *message body length* to the feature improves *SNARE*'s performance noticeably. The ability to identify spammers improves and the false-positive rates drop, irrespective of which classifier is used. This is a satisfying result and it allows for balancing resource availability and responsiveness with performance.

Addition of *number of recipients* to the set of features clearly does not help. It is somewhat expected that this feature will not add any new discrimination power since the number of recipients listed on the "To" field can easily be limited to one by putting the target email addresses on "Cc" and "Bcc" fields. Besides, if the spammers always place a single recipient address in the "To" field, this value is going to be same as the large majority of legitimate messages. This experiment only validates this intuitive fact. Since we did not have logs of additional fields in the SMTP header beyond the count of email addresses on the "To" field, we could not evaluate whether considering number of recipients listed under "Cc" and "Bcc" headers will make this a useful feature to consider.

5.3 Aggregate Features

In this section we evaluate the effectiveness of aggregate features constructed by combining history from multiple messages from a single sender. With the exception of distances to k nearest neighbors, the rest of the features here will be different from the previously evaluate per-message features since values from multiple messages are aggregated. If multiple messages from a sender are available, the following features, in addition to k nearest neighbor distances, can be constructed:

- the mean and variance of geodesic distances, message body lengths and number of recipients.
- the number of messages sent to each of top d most frequently targeted domains. (In our experiments, $d = 100$.)

If we have logs of IP messaging behavior for 24 hours or more, we can also construct:

- a timeseries of the *number of messages* sent in each interval over the course of the day
- a timeseries of the *number of bytes* transmitted in each interval

Table 5 shows the performance of the two classifiers with these aggregate features. There are few interesting observations here; (1) the prediction accuracy obtained using aggregate features is not significantly better than the one earlier

		Confusion Matrices for SVM prediction		Confusion Matrices for decision tree prediction	
		Classified as		Classified as	
		Spam	Ham	Spam	Ham
Multiple Messages	Spam	8456(86.24%)	1349(13.76%)	8781(**89.56%**)	1024(10.44%)
	Ham	564(**5.75%**)	9241(94.25%)	694(7.08%)	9111(92.92%)
		Classified as		Classified as	
		Spam	Ham	Spam	Ham
24+ hour history	Spam	8031(81.91%)	1774(18.09%)	8828(**90.04%**)	977(9.96%)
	Ham	634(**6.47%**)	9171(93.53%)	786(8.02%)	9019(91.98%)

Table 5: *SNARE* performance using aggregate features computed using multiple messages from a single sender

obtained from single message, (2) addition of the time-series features slightly deteriorate the prediction accuracy instead of improving it, (3) the support vector machine now outperforms the decision tree in terms of false-positive rate.

An extremely surprising and counterintuitive result is that *aggregate features do not yield superior performance*, despite incorporating more information. There may be many possible explanations for this. One possible reason is the increase in dimensionality of the feature space and decrease in the number of available training data points. The dimensionality of the feature space now increases five-fold. At the same time, the actual number of training points available to train the classifier using history over a fixed period decreases as multiple log entries are aggregated into a single feature vector. This means that the classifier is being trained in a much higher dimensional feature space using fewer training points. As a result, the classifiers overfitted the training data which was evident from the very high prediction accuracy on training set (close to 99%). Additionally, the target domain distribution and time-series features will also tend to be very sparse for large proportion of the senders. Most of the spammers and legitimate senders will likely not send high volume of messages from a single IP in any given observation period. So, we may not see messages to most of the domains that form part of our feature set. At the same it is infeasible to use message counts to all the target domains that are in the logs as that will only result in even higher dimensional and more sparse feature vectors.

The addition of time-series features causes the prediction accuracy to deteriorate for the similar reasons of sparsity and increased dimensionality. Further, neither of the classifiers used in this study account for the autocorrelation between consecutive bins as they essentially treat each bin as an independent feature. In fact, there no existing classifiers that do this in a reasonable way. Better exploitation of temporal autocorrelation in the two time-series suggested here to improve the prediction accuracy is something we are currently investigating.

Finally, even though the decision tree classifier identifies more spammers correctly, the support vector machine has a better false-positive rate in this case. As such, support vector machine is a better classifier to use with aggregate features.

6. RELATIVE INFLUENCE OF FEATURES

In this section we use the the fact that decision trees produce interpretable rules to understand the *relative importance* of the features we have examined in Sections 3 and 4. Table 6 shows the the six most frequently used prediction rules for each class, accounting for 37.3% of ham data points and 54.6% of spam data points respectively. It is important to remember these rules are just illustrative and don't provide a complete picture of the overall decision tree structure.

Table 6 shows that two features—the geodesic distance between the sender and the recipient and the distances to the k nearest neighbors—play an important role in separating out a large fraction of spammers from good senders. This result is quite promising, since, as per our discussion in Section 5.1.2, these features are also evasion resistant and lightweight (they can be gleaned from a single packet, assuming history about neighboring IP addresses is available).

The rules also show that the classifier accounts for legitimate server groups that might have a sequence of IP addressess next to each other (for example, spam rule 2 and ham rule 1), which might otherwise resemble a group of compromised hosts on a subnet. The spam-to-ham probability also appears among these top rules, demonstrating the fact that incorporating time of the day is helpful in improving prediction. The classifier also captures the fact that the body length of the messages sent by spam senders is usually not very large.

Rule 1 for spammers is remarkable: On one hand, it is interesting that the decision tree generated a rule that appears to be based on a large spam campaign (based on the fact that it is weeding out messages of a common size). On the other hand, it is also a cause for concern, because such rules could result in false positives; such a rule is also easy to evade. In our current system this rule worked well as the corresponding leaf node had a purity of 99.7%, i.e. 99.7% of data points covered by this rule had a spam label. In a deployed system, it would be straightforward to assign weights to the features and adjust the information gain criteria accordingly, so that features that could be more prone to evasion do not play a dominant role in the classification role. Additionally, the class probability at each of the leaf nodes [28] can be used as a confidence score. *SNARE* can provide this confidence

Rules for Spam		Rules for Ham	
1) mesg_size > 4803 mesg_size <= 4819 -> class spam --------------------- # points: 5177 (8.62%) purity : 99.7%	4) geo_dist > 4639.38 nst_nbr1 > 1 nst_nbr2 > 20 nst_nbr20 <= 5532 mesg_size <= 63256 -> class spam ---------------------- # points: 1936 (3.23%) purity : 99.5%	1) geo_dist <= 3930.29 nst_nbr1 <= 1 pr_ratio <= 1.09355 mesg_size > 14990 -> class ham ---------------------- # points: 3160 (5.27%) purity : 99.6%	4) geo_dist <= 2694.95 nst_nbr1 <= 8 nst_nbr4 > 2553 nst_nbr20 > 21805 mesg_size > 121 -> class ham ---------------------- # points: 1462 (2.44%) purity : 99.3%
2) nst_nbr1 > 8 nst_nbr1 <= 65 nst_nbr9 > 95 nst_nbr15 <= 1451 mesg_size <= 4806 -> class spam --------------------- # points: 3233 (5.38%) purity : 99.8%	5) geo_dist > 1739.56 nst_nbr3 > 45 nst_nbr6 <= 1595 mesg_size <= 369 -> class spam ---------------------- # points: 1490 (2.48%) purity : 99.7%	2) geo_dist <= 3930.29 nst_nbr2 <= 132 nst_nbr12 > 1696 mesg_size > 9158 -> class ham ---------------------- # points: 2077 (3.46%) purity : 99.3%	5) geo_dist <= 2198.92 nst_nbr2 > 3942 pr_ratio <= 0.89066 mesg_size > 5762 -> class ham ---------------------- # points: 1418 (2.36%) purity : 99.4%
3) geo_dist > 2694.95 nst_nbr16 <= 1883 mesg_size <= 5857 -> class spam --------------------- # points: 3064 (5.11%) purity : 99.7%	6) nst_nbr1 > 8 nst_nbr17 <= 88787 mesg_size > 46 mesg_size <= 137 -> class spam ---------------------- # points: 1476 (2.46%) purity : 98.7%	3) nst_nbr16 > 13136 mesg_size > 61750 -> class ham ---------------------- # points: 1668 (2.78%) purity : 99.8%	6) geo_dist <= 3930.29 nst_nbr20 > 33 prob_ratio <= 1.02114 mesg_size > 9158 -> class ham ---------------------- # points: 1413 (2.36%) purity : 98.6%

Glossary:

geo_dist = Geodesic distance in miles

nst_nbr$k$ = distance to the k^{th} nearest neighbour in IP space

# points = number of training points covered by the rule

mesg_size = Message Size in bytes

pr_ratio = Probability ratio of spam to ham at the time of day

purity = % of points in the leaf node from the target class

Table 6: Six most frequently used classification rules for each class extracted from the decision tree

score to the mail server which can then use its own thresholds to decide whether to block the IP straightaway or subject it to further scrutiny.

7. VALIDATION AGAINST BLACKLIST

We performed a preliminary validation against IP addresses listed in the SpamHaus blacklist to investigate whether some of *SNARE*'s misclassifications are in fact additional spammers that *SNARE* had caught. We used the Secure Computing data from October 22 and queried the 19,610 IP addresses from the training set with aggregrate features. Of these, 9,805 had a high spam score in the Secure Computing data. SpamHaus blacklist data from October 1, 2007 to May 8, 2008 was used for this lookup. Out of the original 9,805 IP addresses, 6,508 were listed in the SpamHaus blacklist by October 22, and an additonal 101 were added after that date. The SVM classifier misclassified 564 of the senders labelled as legitimate senders in the training data. We queried these misclassified IPs against the Spamhaus blacklist and discovered that 11 of these were actually listed in the SpamHaus blacklist in October. Two of these were removed and then added back into the blacklist after October 22nd. Nevertheless, these low numbers suggest that either SpamHaus continues to miss some of the IP addresses that *SNARE* labeled as spam, or our false positives are in fact misclassifications.

8. RELATED WORK

SNARE is motivated by recent studies that have helped understand the network-level behavior of spammers, botnets, dynamically assigned IP space and the relationship among these. It is also closely related to systems for content independent spam filtering and sender reputation determination. We discuss some the related works for each area below.

Characterization studies. Recent characterization studies have provided increasing evidence that spammers' have some distinct network-level behavioral patterns and helped strengthen the case for devising techniques for automated sender reputation system using network behavioral patterns. Ramachandran *et al.* [23] used data from a single spam trap and showed that spammers utilize transient botnets to spam at low rate from any specific IP to any domain. Xie *et al.* [27] discovered that a vast majority of mail servers running on dynamic IP address were used solely to send spam and accounted for more than 42% of all emails received at Hotmail. In their recently published study [26], they demonstrate a technique to identify bots by using signatures constructed from urls in spam messages. They also showed that emails from a botnet are often sent in a highly synchronized fashion and that hosts within the same botnet exhibit similar sending patterns. Their signature based botnet identification is different from *SNARE*'s reputation system in that it uses the message content whereas *SNARE* does not use any content. Pathak *et al.* [19] deployed a relay sinkhole to gather data from multiple spam senders destined for multiple domains. They used this data to demonstrate how spammers utilize compromised relay servers to evade detection. They also showed the separation between the high volume standalone spammers and low volume senders that are coordinated with each other. While this work also used data from multiple vantage points, the data is somewhat different from ours since it was collected from a sinkhole and consisted mostly of spam messages.

Sender reputation. The SpamTracker [23] system is most closely related to *SNARE* in using network level behavioral features from data aggregated across multiple domains to build a behavioral reputation system. While that work initiated the the idea of *behavioral blacklisting*, we have discovered in the course of designing *SNARE* that there are several other behavioral features that are better suited for separating spammers from legitimate senders, when compared to the *distribution of messages across recipient domains* that was used as a feature in SpamTracker. Tang et al. explored the detection of spam senders by analyzing the behavior of IP addresses as observed by query patterns [25]. Their work focuses on feedback retrieved from IP address lookups by analyzing the breadth and the periodicity of message volumes in relation to sources of queries. There are also several commercially deployed IP reputation systems that are similar in spirit to *SNARE* that are part of a bigger spam filtering system. For example Secure Computing [2] and IronPort [1] deploy spam filtering appliances to hundreds or thousands of domains which then query the central server for sender reputation and also provide meta-data about messages they receive. As we noted earlier our data is in fact obtained from these kind of logs gathered by the Secure Computing servers and we intend to ultimately deploy *SNARE* as part of Secure Computing's TrustedSource[3] sender reputation system.

Non-content Spam Filtering. The Trinity system [9] is designed to be a distributed, content-free spam detection system for messages originating from botnets. It uses message count volumes to determine whether an email could have originated from a bot. This system can potentially be used as component of *SNARE* to provide additional features.The SpamHINTS project [13] also has the stated goal of building a spam filter using analysis of network traffic patterns instead of the message content. Clayton's earlier work on *extrusion detection* involves monitoring of server logs at both the local ISP[11] as well as the remote ISP [12] to stop relaying of spam messages and detect spam senders. While this work has similar objectives as ours, the methods proposed under spamHINTS focus more on properties related to SMTP sessions from a single sender.

9. CONCLUSION AND FUTURE WORK

Although there has been much progress in content-based spam filtering, state-of-the-art systems for sender reputation are relatively unresponsive, incomplete, and coarse-grained. Towards improving this state of affairs, this paper has presented *SNARE*, a sender reputation system that can quickly, accurately, and automatically classify email senders based on features that can be determined early in a sender's history—sometimes after seeing only a single IP packet. *SNARE*'s classification algorithms achieve 93% detection rate and only a 7% false-positive rate when looking at only a single IP packet from a sender, without having any prior history about the sender. Using additional lightweight features, *SNARE* can detect about 94% of spam with only a 5% false positive rate. These results suggest that *SNARE* might ultimately be used as a sender reputation system that is used as a first-pass filter for email filtering systems. The false-positive rate may still likely be too high to be used in some practical systems; therefore, a future challenge will be to further reduce false positives, or to figure out how to incorporate *SNARE* into an email system where such a false positive rate is tolerable (e.g., one that forces senders to retry, delays all suspected spam, etc.).

In addition to presenting a first-of-its kind automated classifier based on early, evasion-resistant features, this paper has presented several additional contributions. First, in Section 3, we presented a detailed study of various spatial and temporal characteristics of both spammers and legitimate senders, as well as an examination of how well each feature distinguishes spammers from legitimate senders. Second, in Section 6, we presented a detailed analysis of the relative importance of each of the features we studied. Our results are surprising and encouraging: Some of the most important features for distinguishing spammers from legitimate senders are those that can be gleaned from a single packet. This result was surprising because intuition would suggest that more information would produce a better classifier. This result was also very encouraging: it suggests that the strongest features for distinguishing spammers from legitimate senders may, in fact, be the most lightweight and robust ones as well.

10. REFERENCES

[1] Ironport. http://www.ironport.com.

[2] Secure Computing. http://www.securecomputing.com.

[3] Secure Computing TrustedSource. http://www.trustedsource.org.

[4] Sorbs: Spam and open relay blocking system. http://www.au.sorbs.net/.

[5] Spamcop. http://www.spamcop.net/bl.shtml.

[6] Spamhaus ip blocklist. http://www.spamhaus.org.

[7] D. Alperovitch, P. Judge, and S. Krasser. Taxonomy of email reputation systems. In *Proc. of the First International Workshop on Trust and Reputation Management in Massively Distributed Computing Systems (TRAM)*, 2007.

[8] P.O. Boykin and V. Roychowdhury. Personal Email Networks: An Effective Anti-Spam Tool. *IEEE Computer*, 38(4):61–68, 2005.

[9] A. Brodsky and D. Brodsky. A distributed content independent method for spam detection. *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets table of contents*, pages 3–3, 2007.

[10] C.J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[11] R. Clayton. Stopping spam by extrusion detection. In *First Conference of Email and Anti-Spam (CEAS)*, 2004.

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

[13] R. Clayton. spamhints: Happily it's not the same. www.spamhints.org, 2007.

[14] J. Golbeck and J. Hendler. Reputation network analysis for email filtering. *Proceedings of the First Conference on Email and Anti-Spam*, pages 30–31, 2004.

[15] J. Goodman, G.V. Cormack, and D. Heckerman. Spam and the ongoing battle for the inbox. *Communications of the ACM*, 50(2):24–33, 2007.

[16] E. Hulton and J. Goodman. Tutorial on junk email filtering. *ICML*, 2004.

[17] T. Joachims. Making large-Scale SVM Learning Practical. Advances in Kernel Methods-Support Vector Learning, B. Schölkopf and C. Burges and A. Smola. *MIT-Press*, 1:999, 1999.

[18] LLC MaxMind. GeoIP API. http://www.maxmind.com/app/api, 2007.

[19] A. Pathak, Y. Hu, C., and M. Mao, Z. Peeking into spammer behavior from a unique vantage point. In *First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '08)*, 2008.

[20] JR Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[21] J.R. Quinlan. *C4. 5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[22] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *Proceedings of the ACM SIGCOMM*. ACM Press New York, NY, 2006.

[23] A. Ramachandran, N. Feamster, and S. Vempala. Filtering spam with behavioral blacklisting. In *ACM Conference on Computer and Communications Security*. ACM Press New York, NY, 2007.

[24] H. Shimazaki and S. Shinomoto. A Method for Selecting the Bin Size of a Time Histogram. *Neural Computation*, 19(6):1503, 2007.

[25] Y. C. Tang, S. Krasser, P. Judge, and Y.-Q. Zhang. Fast and effective spam IP detection with granular SVM for spam filtering on highly imbalanced spectral mail server behavior data. In *Proc. of The 2nd International Conference on Collaborative Computing*, 2006.

[26] Y. Xie, F. Yu, , K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov. Spamming bots: Signatures and characteristics. In *Proceedings of the ACM SIGCOMM*, 2008.

[27] Y. Xie, F. Yu, K. Achan, E. Gilum, M. Goldszmidt, and T. Wobber. How dynamic are ip addresses. In *Proceedings of the ACM SIGCOMM*, 2007.

[28] B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 609–616, 2001.