Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2008-008 February 15, 2008

Exploiting Transport-Level Characteristics of Spam

Robert Beverly and Karen Sollins

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Exploiting Transport-Level Characteristics of Spam

Robert Beverly
MIT CSAIL
rbeverly@mit.edu

Karen Sollins
MIT CSAIL
sollins@csail.mit.edu

ABSTRACT

In the arms race to secure electronic mail users and servers from unsolicited messages (spam), the most successful solutions employ techniques that are difficult for spammers to circumvent. This research investigates the *transport-layer* characteristics of email in order to provide a new, novel and robust defense against spam. We find that spam SMTP flows exhibit TCP behavior consistent with traffic competing for link access, large round trip times and resource constrained hosts. Thus, SMTP flow characteristics provide sufficient statistical power to differentiate between spam and legitimate mail (ham). We build "SpamFlow" to learn and exploit these differences. Using machine learning feature selection we identify the most discriminatory flow properties and effect greater than 90% spam classification accuracy without content or reputation analysis. SpamFlow correctly identifies 78% of the false negatives generated by a popular content filtering application – demonstrating the power in combining SpamFlow with existing techniques. Finally, we argue that SpamFlow is not easily subvertible due to economic and practical constraints inherent in sourcing spam.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design; H.4.3 [**Information Systems Applications**]: Communications Applications

Keywords

spam, transport behavior, SMTP, security, machine learning, prediction

1. INTRODUCTION

By all estimates, unsolicited email (spam) is a pressing and continuing problem on the Internet. A consortium of service providers reports that across more than 500M monitored mailboxes, 75% of all received mail is spam, amounting to more than 390B spam messages over a single quarter [9]. Spam clogs mailboxes, slows servers and lowers productivity. Not only is spam annoying, it adversely affects the reliability and stability of the global email system [1].

Current methods for mitigating spam include content analysis [8, 14], collaborative filtering [17, 10], reputation analysis [18, 16], and authentication schemes [2, 20]. While these methods are effective, none offer a panacea for preventing spam. Unfortunately, spam is an arms race where spammers quickly adapt to the latest prevention techniques.

We propose a fundamentally different approach to identifying spam that is based on two observations. First, spam has an extremely low penetration rate due to filtering and user ambivalence. Therefore spammers must send extremely large volumes of mail to remain commercially viable. Second, spammers increasingly rely on zombie "botnets," [4] large collections of compromised machines under common control, as unwitting participants in obfuscating the origin of spam [6]. Botnet hosts are often widely distributed with low, asymmetric bandwidth connections to the Internet. Combining these observations we make the following hypothesis: the network links and hosts which source spam are constrained. Specifically, we ask whether *the transport level characteristics of email flows provide sufficient statistical power to discriminate between spam and valid email.*

In investigating this hypothesis, we first gather a live data set and examine a variety of SMTP (Simple Mail Transport Protocol) [7] flow characteristics in detail. Based on the statistical discriminatory power of these flows, we develop SpamFlow. Our major contributions are:

1. Observations of significant differences between the probability distributions of spam and ham for particular flow features. E.g. 99% of ham flows, compared to only 24% of spam flows, have an RTT $\leq$ 100ms.
2. SpamFlow, a classifier that leverages statistical differences between flows to achieve $>$ 90% accuracy, precision and recall without content or reputation analysis.
3. Correctly identifying 78% of the false negatives generated by SpamAssassin [8]. Thus, SpamFlow may usefully be combined with existing techniques as a weighted vote.

Consequently, we hope this paper serves to identify a new area of spam research and presents a working system which sources of spam cannot easily evade.

2. EXTRACTING INSIGHT FROM MAIL TRANSPORT

Is it reasonable to believe that a spammer's TCP/IP traffic characteristics are sufficiently different than traffic from Mail Transport Agents (MTAs) sending legitimate mail? Our initial impetus to understand the transport behavior of spam came from observing seemingly anonamlous behavior of TCP sockets on our local MTA.

2.1 Socket Behavior

To build intuition on the real-world behavior of SMTP connections, we manually investigate several sockets on our mail server. We observe many sockets that remain in particular TCP states for an unexpectedly long time. The corresponding traffic traces are revealing:

SYN_SENT: Many sockets remain in the SYN_SENT state for an extended period of time, where the remote server sends a SYN, but fails to acknowledge our MTA's response. These connections may be the result of scans for mail servers or indicative of congestion near the source.

LAST_ACK: Frequently, sockets on our MTA remain stuck in the LAST_ACK TCP state. Manual investigation of the packets show senders that successfully open a TCP connection and send SMTP commands. However, these servers do not reliably receive our MTA's acknowledgments, leading to many retransmissions. In many instances, the remote server eventually closes its connection by sending a FIN. Our server acknowledges the FIN and closes its own connection. However, the remote server never sends an acknowledgment leading to our MTA remaining in LAST_ACK. Appendix (I) provides a detailed example trace of this situation.

FIN_WAIT1: Similarly, our MTA has many sockets stuck in state FIN_WAIT1. Inspecting these sockets, we find remote servers that establish a TCP connection and again have trouble responding with acknowledgments. The remote server never sends any data and our MTA eventually times out, closing the TCP connection. However, the remote host has seemingly disappeared. Since the remote server does not acknowledge the FIN our MTA sends on the connection, the socket stays in FIN_WAIT. Appendix (II) provides a detailed trace of traffic that leads to this situation.

Clearly, the traffic for the connections which we manually investigate is suggestive of abnormal behavior or resource exhaustion. To better understand general behavior in a systematic way, we instrument an MTA to collect passive flow data for the email messages it receives.

2.2 Data Collection

Figure 1 depicts our collection methodology. Let the server's IP address be S. We passively capture and timestamp all SMTP packets including the TCP and IP headers. SMTP sessions are processed by the MTA which then uses SpamAssassin (SA) to label each message as spam or ham. Our server is connected to the local network via a non-congested 100Mbps Ethernet which is in turn connected to the wide area Internet via multiple diverse Gigabit speed links.

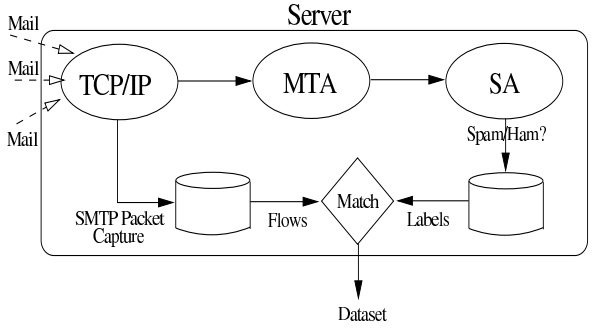


Figure 1: Data collection: incoming SMTP packets are captured and coalesced into flows. Each flow is matched with a binary spam or ham label.

We coalesce the captured email packets into flows. Define a flow $F_{(IP:Port)}$ as all TCP packets $(IP : Port) \rightarrow (S : 25)$ and $(S : 25) \rightarrow (IP : Port)$. Thus, we are interested in undirected SMTP TCP connections between remote hosts and our MTA[1]. Each email message is matched with its corresponding SMTP flow. However, MTAs often receive many connections from the same remote host. To avoid the ambiguity in synchronizing messages in time, particularly when aggressive MTAs initiate multiple simultaneous connections, we modify our MTA server (S) to include the incoming connection's source port number in each mail header[2]. When receiving many, potentially simultaneous, emails from the same source, the IP and port number in the header allow us to match the message to its flow.

Over the course of one week in January, 2008, we collect a total of 18,421 messages for a single mail account, 220 of which were legitimate while the remaining 18,201 were spam (98.8%). Of the ham messages, 39 were from unique mail domains. Note that while we use SpamAssassin to separate messages, SpamAssassin can generate false-negatives and false positives. We manually relabel 127 (0.7%) incorrectly marked messages. We discuss performance differences between SpamFlow and SpamAssassin in §4. Even though our data set is limited in scope, it reveals surprising and evocative characteristics of spam in contrast to ham.

2.3 An Example

We manually examine two emails (one spam, one ham) from our data to provide further intuition. Figure 2 shows our MTA's round trip time (RTT) estimation for each of the two

[1]Since our server's IP and SMTP port are fixed ($S : 25$), these fields are not included in the flow tuple.
[2]In fact, we use the environment variables provided by `tcpserver`, an `inetd` replacement.

(a) Received: from vms044pub.verizon.net
From: "Dr. Beverly, MD" <b@ex.com>
Subject: thoughts

Dear Robert, I hope you have had
a great week!

(b) Received: from unknown (59.9.86.75)
From: Erich Shoemaker <ried@ex.com>
Subject: Repl1ca for you

A T4g Heuer w4tch is a luxury statement on its
own. In Prest1ge Repl1cas, any T4g Heuer...

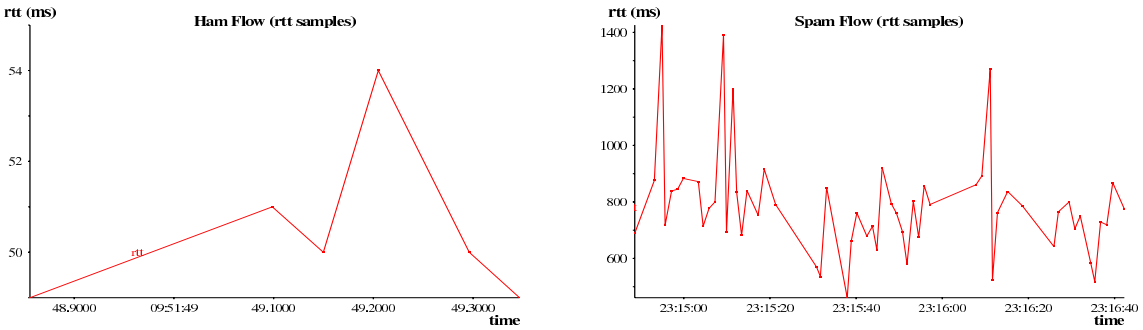


Figure 2: Comparison of (a) ham and (b) spam flow RTT estimation. Note that the RTT variance differs by an order of magnitude - an indication of flow congestion effects.

different emails. Figure 2(a) is a valid message; the RTT estimate is low and more importantly shows little variation across the flow's duration. In contrast, Figure 2(b) depicts a spam mail and flow RTT estimate. Not only is the RTT high, it varies significantly with a minimum less than 500ms and a maximum nearly 1.5s. Such wildly varying RTT estimates are often indicative of persistent congestion and is one indirect sign that a flow may belong to a spammer.

2.4 Transport Characteristics

Figure 3 compares the RTT, maximum idle time and number of FIN packets between ham and spam for our entire data set. Here we define the RTT as the initial RTT estimate inferred from the three-way TCP handshake. Figure 3(a) shows the cumulative distribution of RTT times in our data. The difference between spam and ham is evident. While more than 20% of ham flows have an RTT less than or equal to 10ms, almost no spam flows have such a small initial RTT. The RTT of nearly all ham flows is 100ms or less. In contrast, 76% of spam flows have an RTT greater than 100ms.

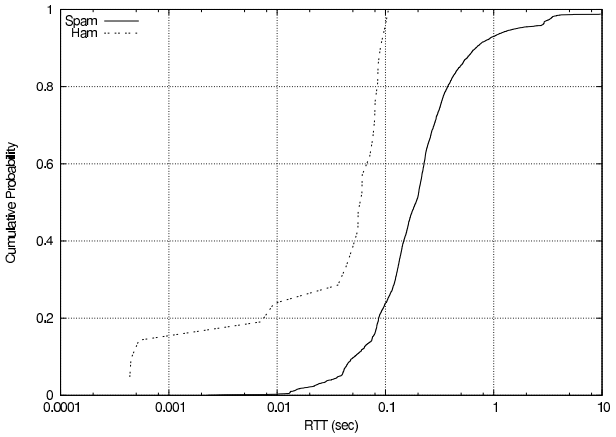
In order to show how a feature such as RTT can be used to provide a classifying discriminator, we show the posterior probability of an email being a spam, given that the RTT of the message (rtt) is greater than x. We use Bayes' rule to take the causal information and form a diagnosis:

$$P(spam|rtt > x) = \frac{P(rtt > x|spam)P(spam)}{P(rtt > x)} \quad (1)$$

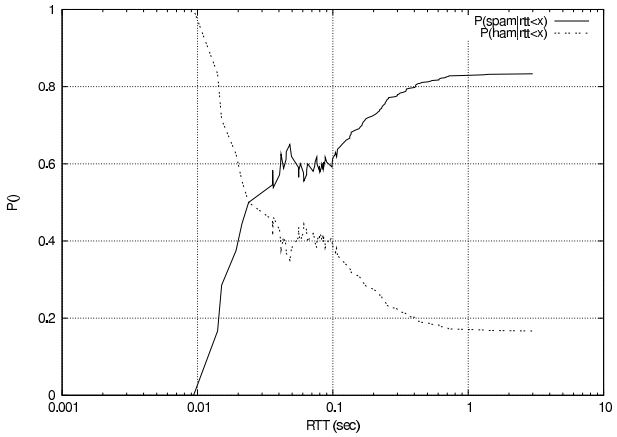
Figure 3(b) shows the conditional probability of a spam message across a continuous range of RTTs. We include the probability of a ham message in the figure as well; these probabilities sum to one, hence providing mirror images of each other. With an RTT less than 10ms, the probability is strongly biased toward being a ham message. In the range $[0.02, 0.1]$s, the probability estimate is relatively neutral without a strong bias toward either category. However, after 100ms, there is a strong tendency toward the message being spam. This conditional probability distribution corresponds exactly to the data in Figure 3(a).

The differences in RTT raise several interesting points. For some classes of users, it is not unexpected that legitimate email originates from geographically nearby sources. Thus, it is prudent in many cases to take advantage of locality of interest. RTT may be less of a distinguishing characteristic though for users with frequent trans-continental conversations. However, approximately 50% of the spam messages have an RTT greater than 200ms, suggesting that the remote machines are quite remote, overloaded or reside on constrained links. Further, the $\sim 10\%$ of flows with an RTT greater than one second cannot easily be explained by geographic distance and are more likely evidence of persistent congestion or end host behavior. We emphasize that RTT is just one potential feature. In instances where users receive legitimate email with large RTTs, the system may use a threshold strategy or simply lower the relative importance of RTT in favor of other flow features. Just as content filters are frequently customized per-user, the distinguishing flow characteristics can be unique to each user based on his or her receiving patterns.

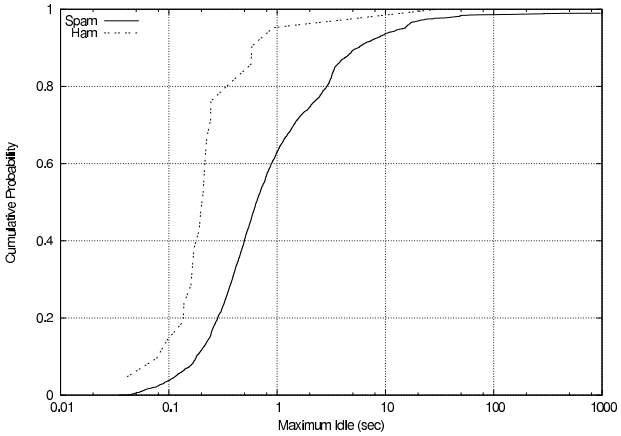
As a second feature, consider maximum idle, the maximum



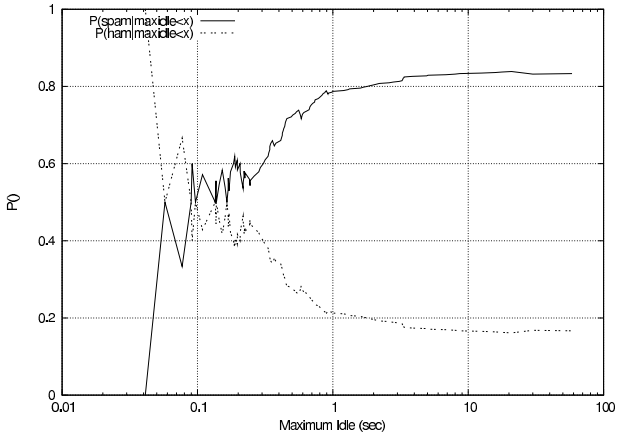
(a) RTT Cumulative Probability Distribution



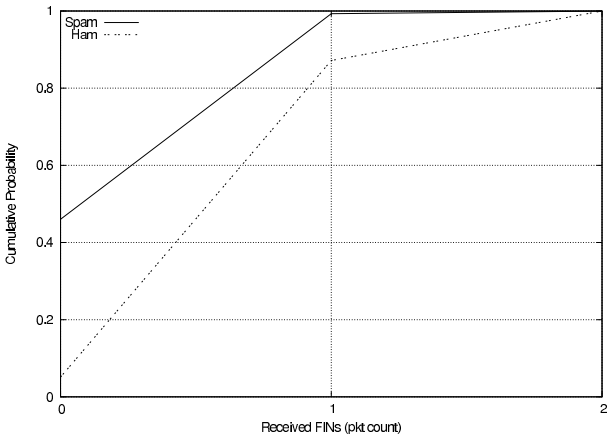
(b) RTT Conditional Probability



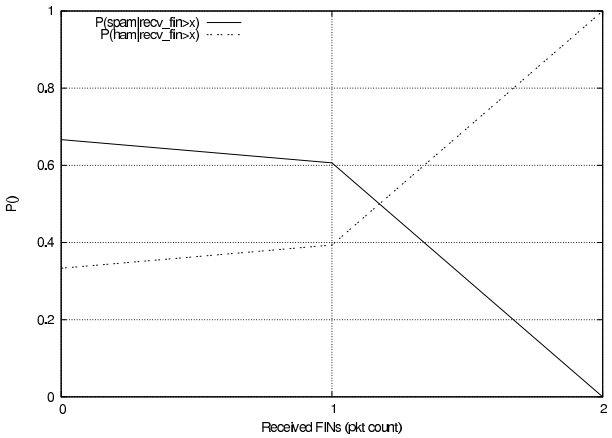
(c) Maximum Idle Time Cumulative Probability Distribution



(d) Maximum Idle Time Conditional Probability



(e) Received FIN Count Cumulative Probability Distribution



(f) Received FIN Count Conditional Probability

Figure 3: Comparing spam and ham probability distributions for RTT, idle time and received FIN count (left column). The resulting conditional probability distributions (right column) serve as a discriminator.

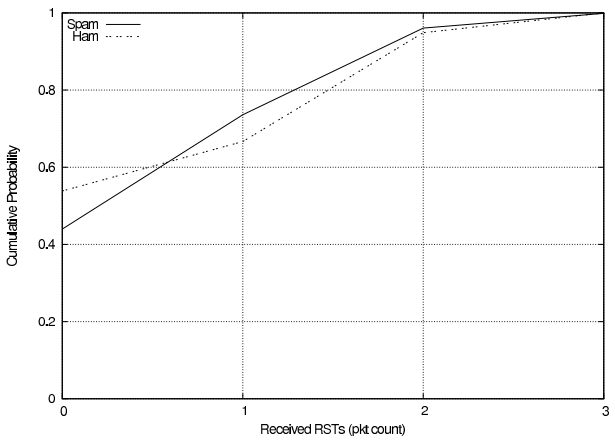


Figure 4: Non-features: distribution of received TCP RST count is similar for spam and ham. Surprisingly, this feature provides little discrimination.

time interval between two successive packets from the remote MTA. In some instances the maximum idle time directly reflects the initial RTT, but is often different. Figure 3(c) depicts the cumulative distribution of maximum idle times. Again, we see marked differences between the character of spam and ham flows. For instance, nearly 40% of spam flows have a maximum idle time greater than one second, events unlikely due to geographical locale. Figure 3(d) shows the conditional probability that the message is spam. After a maximum idle of 250ms, the probability tends strongly toward spam, as there are few legitimate messages with such a long idle time.

Finally, to emphasize that there are many potential features available in a flow, we examine TCP FIN segments. In a normal TCP session termination, each endpoint issues a finish (FIN) packet to reliably end the connection. Figure 3(e) shows that almost 45% of the spam email flows do not send a FIN compared to only 5% for ham. Finally, a small fraction of ham flows result in two FINs whereas only 0.7% of spam flows send more than one FIN. The resulting conditional probabilities are given in Figure 3(f). Section 3 details all of the features we consider.

2.5 Non-features

A strength of a statistical approach is in systematically identifying not only good features, but also poor features. Several flow properties we initially expected to be a strong indication of spam provide little differentiation. For example, one might expect ill-behaved flows to tear down the TCP connection using TCP resets (RSTs) rather than a graceful shutdown with FIN packets. However, as Figure 4 demonstrates, the distribution of received RSTs is very similar between spam and ham. Surprisingly, only 53% of ham flows contain no reset packets while 28% contain two RSTs.

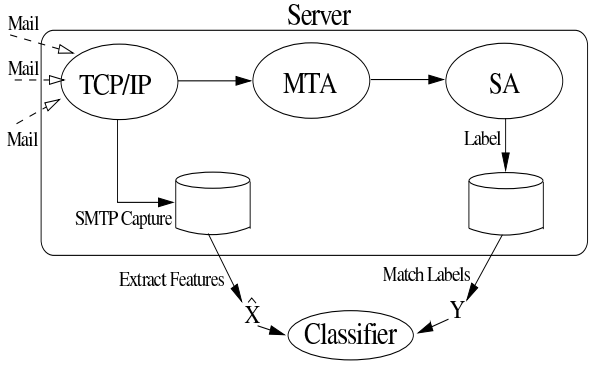


Figure 5: Supervised classification methodology: learning features $\hat{X}$ are extracted from captured SMTP flows. SpamAssassin generates training labels (Y) for each flow based on the email.

Manual investigation of the data reveals that many MTAs, including those of popular web mail services such as Google and Yahoo, send RST packets after sending the SMTP quit command. Detailed traces of this phenomenon are provided in the Appendix (III-IV). Frequent RSTs are not limited to a few select large servers. In addition, the source code of the popular Postfix [5] MTA software contains the SO_LINGER socket option that causes it to perform an abortive close and generate TCP reset packets.

In all, the preceding analysis provides evidence that spam and ham flows are sufficiently different to reliably distinguish between them. The important point of note is that we examine neither the content nor origins of incoming emails. Instead our determination of an email's legitimacy is based entirely upon the incoming flow's *transport characteristics*.

3. EXPERIMENTAL METHODOLOGY

The intuition behind our scheme is simple. Because spammers must send large volumes of mail, they parallelize and asynchronously transmit mail continuously. In addition, the sources of spam are frequently large compromised "botnets," which not only are resource constrained, but are typically connected to the Internet by links with asymmetric bandwidths, e.g. aDSL and cable modems. Therefore the flows that comprise spam TCP traffic exhibit behavior consistent with traffic competing for link access. A spammer's traffic is thus more likely to exhibit TCP timeouts, retransmissions, resets and highly variable round trip time (RTT) estimates as seen in §2.

To validate our hypothesis, we devise the experimental methodology in Figure 5. Incoming emails arrive at our MTA as SMTP TCP packets. The captured SMTP packets are coalesced into flows. Properties of each flow (F_i) provide the learning features ($\mathbf{X}_i$). For each F_i, the corresponding email message is processed by SpamAssassin (SA). The binary label from SpamAssassin provides the ($Y_i \in \{\pm 1\}$) label to

Table 1: Flow properties used as classification features

Feature	Description
Pkts	Packets
Rxmits	Retransmissions
RSTs	Packets with RST bit set
FINs	Packets with FIN bit set
Cwnd0	Times zero window advertised
CwndMin	Minimum window advertised
MaxIdle	Maximum idle time between packets
RTT	Initial round trip time estimate
JitterVar	Variance of interpacket delay

perform supervised learning. We emphasize that our use of SA is only to label training data; we in no way use SA to perform classification on new flows. Further, as stated previously, we manually inspect the results of SpamAssassin's decisions to remove false positives and negatives.

We develop a flow extractor routine that identifies relevant features of each flow. Currently we extract the features in Table 1. Note that while our flows are undirected, particular features have direction. For instance, we distinguish between received and sent packet counts, RSTs, FINs and retransmissions. Thus, including separate directional features for each of the properties, we consider 13 features in total for each flow.

Our feature extraction step thus provides an input vector $\mathbf{X}_i^d; X \in \mathbb{R}, d = 13$ for flow F_i. From these features, we wish to determine which provide the most discriminative power in picking out spam and how the number of training examples affects performance.

3.1 Data Normalization

In this study, we use only the 39 unique ham mails so that our learning algorithm does not hone in on domain specific effects. For instance, if a majority of email arrives from Yahoo and Google MTAs, the primary features may reflect specific properties of flows from these servers. While nothing precludes learning on the basis of multiple mail flows from a single domain, we seek to understand the generality of SMTP flow characteristics. Our results would likely improve given additional training data on the same domain and MTAs. We defer a full investigation of including mail MTA frequency for later work.

As a result, our data set contains many more spam messages than legitimate messages. To prevent a large discrepancy in the complexion of training samples, we limit our data set to include only five times as many spam messages as valid messages. In each experiment, we select a random set of spam messages that is no more than five times larger than our ham corpus. Thus, the experiments include 39 valid emails and 195 randomly selected spam emails (234 total labeled messages and corresponding SMTP packets).

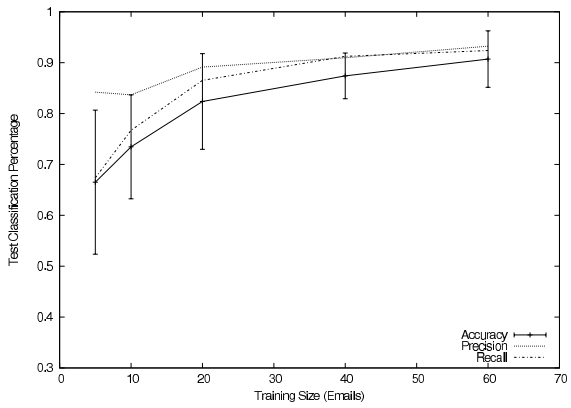


Figure 6: Classification accuracy, precision and recall vs. training size

4. RESULTS

With our data set as described in §2, we train a supervised classifier. In each experiment, we separate the $(\mathbf{X}_i, Y_i)$ pairs from the feature extraction of §3 into a training and test set. We perform a horizontal concatenation of the $\mathbf{Y}$ labels and n x f feature matrix $\mathbf{X}$ to form $\mathbf{D} = [\mathbf{Y}^T : \mathbf{X}]$. To ensure generality, we randomly permute rows of $\mathbf{D}$ for each experiment and run each experiment ten times. For a permuted $\mathbf{D}$, the training data consists of the first i rows of $\mathbf{D}$ while the test set is formed from the remaining $n - i$. In this way the training and test samples are different between experiments.

We use a Support Vector Machine (SVM) classifier [13], but our methodology is independent of the exact learning algorithm. In future work, we plan to examine other classification techniques.

4.1 Performance

We wish to determine the classifier performance and sensitivity to training size. Figure 6 shows the classification accuracy, precision and recall as a function of the training size. For accuracy, we include standard deviation error bars. We achieve approximately 90% accuracy using 60 training emails and more than 80% accuracy with only 20. This accuracy rate is relatively insensitive to the size of the data set, for instance if we include only twice as many spam as valid messages. However, the accuracy rate is misleadingly high if we include all spam messages since the classifier need only guess "spam" to achieve high accuracy. Thus, we also include recall, or the true positive rate and precision measures. Recall is the ratio of true positives to the number of actual positives, $recall = TP/(TP + FN)$ and is therefore a proxy for the number of false negatives. Precision is most important in this application where the majority of

messages are spam. Precision is the ratio of true positives to all predicted positives, $precision = TP/(TP+FP)$, providing a metric of false positives. We see that at 40 training mails, the precision is more than 90%, corresponding to an average of two false positives per iteration. Of note is that the standard deviation is tighter as the number of training emails increases.

We achieve this level of performance using a fourth degree polynomial kernel without any SVM tuning or care in the input feature space. The current false positive rate is higher than is ideal for our application. With further effort, we can likely achieve higher performance. However, we envision our classification scheme as an additional metric in an overall decision tree in just the same way modern filters use multiple tests to form a spam decision.

4.2 Feature Selection

Of the features we extract in Table 1, it is interesting and computationally attractive to determine which provide the most discrimination power. To find these, we turn to feature selection methods [21]. Greedy forward fitting (FF) requires computing a combinatorial number of possible feature combinations. However forward fitting effectively eliminates features that themselves are closely dependent. Often two features individually provide significant power, but the second feature provides little additional classification power. For example, the RTT and maximum idle time may be highly correlated. Forward fitting will continually seek the next best performing feature without this potential dependence.

Forward fitting feature selection simply finds, in succession, the next single feature j that minimizes an error function $V(\cdot)$. Therefore, training error decreases monotonically with the number of features. Figure 7 provides the basic intuition behind feature selection.

Feature selection proceeds in rounds. Formally, in round i let $\mathbf{S}^j$ be an f x i binary selection matrix with $s_{j,i} = 1$ and $s_{k \neq j,i} = 0$. The $i-1$ columns of $\mathbf{S}^j$ are set in previous rounds. Recall that the data $\mathbf{X}$ is an n x f matrix containing n emails with f features each. Let:

$$\mathbf{Z}^j = \mathbf{X}\mathbf{S}^j \tag{2}$$

Thus, $\mathbf{S}^j$ selects feature j in round i. Let $F = \{1 \ldots f\}$ indicate the set of all possible features. We denote θ^i as the set of best features in round i. Then, for a prediction function $f(\cdot)$ and error function $V(\cdot)$, find:

$$\underset{j \in F - \theta^{i-1}}{\operatorname{argmax}} \; V(f(\mathbf{Z}^j), \mathbf{Y}) \tag{3}$$

The feature that best minimizes the error in round i is j, so we update the set of best features: $\theta^i = \theta^{i-1} + j$. Training error is typically an effective proxy for test error. We use SVM training accuracy as the error function although forward fitting can be used with any model and error function.

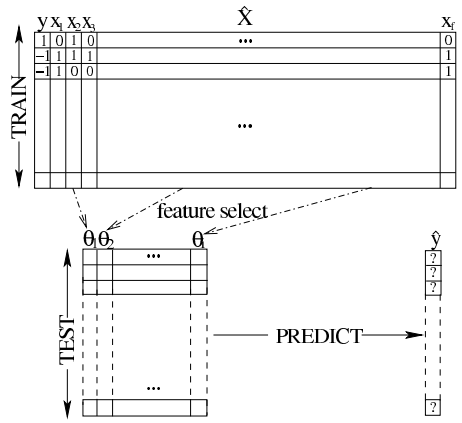


Figure 7: Forward fitting finds a set of features $|\boldsymbol{\theta}| < f$ that provide the least training error. These features are then used in test prediction.

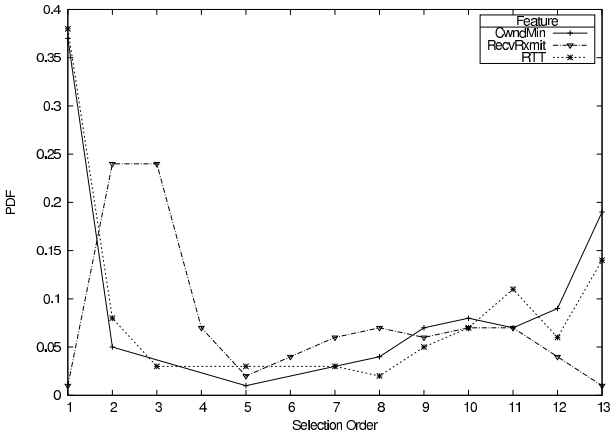
Figure 8 shows the cumulative probability distributions of the selection order for each feature. We split the results into two plots only to improve readability. Figure 8(a) illustrates that both RTT and CwndMin are the most likely features to be selected first, each with approximately 40% probability. Maxidle has around a 10% chance of being the first selected feature and the other features comprise the remaining 10%. In other words, if the learner were given the choice of one and only one feature with which to classify, the learner would choose RTT or CwndMin. RecvRxmit and SentRxmit are not chosen as the first or second features often, but frequently serve as the third and fourth best features.

Figure 8(b) gives the secondary features, those that are more likely to be chosen fifth or later in the order. These features include the RecvFIN, SentFIN, Cwnd0 and JitterVar.

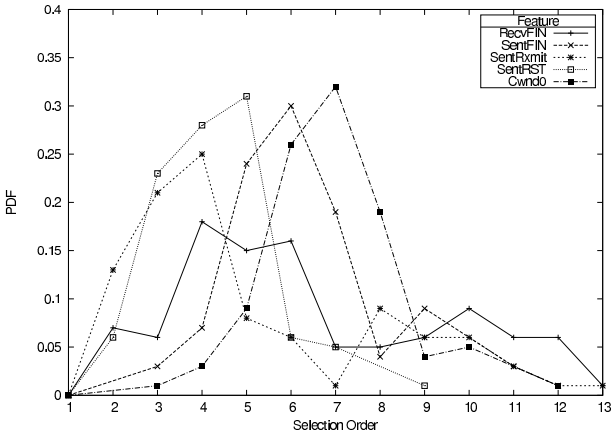
To leverage the results of feature selection, we measure the prediction dependence on the number of best features. Figure 9 gives the results of performing forward fitting, mutual information and random features in each round to select a given number of best features. We include random features to provide a useful baseline. As expected, the random features perform the worst, yet still yield 60-70% accuracy. Forward fitting achieves much higher accuracy, precision and recall, but suffers from over-fitting as the number of features is increased beyond five.

5. RELATED WORK

Current best practices for defending against spam are multi-pronged with four main techniques: content filters, collabo-

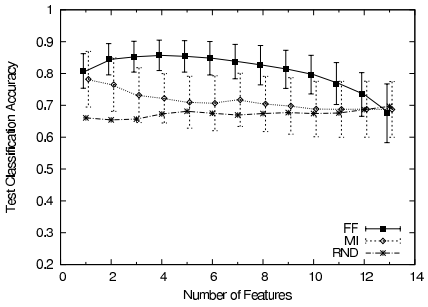


(a) Primary features: minimum congestion window and initial RTT are frequently the best feature; received retransmits are a strong second feature.

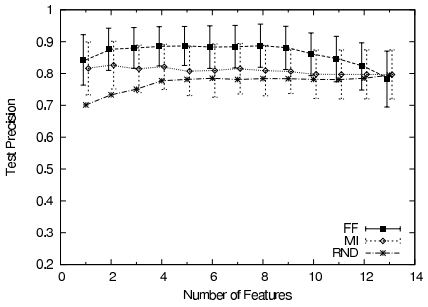


(b) Secondary features: have probability centered in the middle of the selection order indicating flow properties that distinguish ham and spam well.

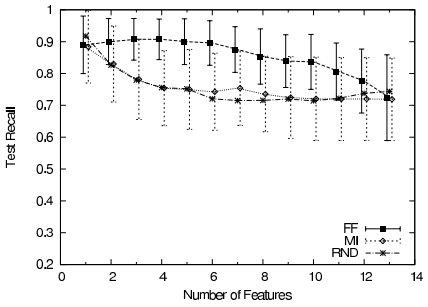
Figure 8: Feature selection order probability distributions demonstrate the relative discriminatory strength of different flow properties.



(a) Prediction Accuracy



(b) Prediction Precision



(c) Prediction Recall

Figure 9: Prediction performance relative to the number of features for Forward Fitting (FF), Mutual Information (MI) and Random (RND) feature selection with an SVM model. The points represent average performance across nodes in our data set, while the error bars show the standard deviation.

rative filtering, reputation systems and authentication schemes. The most successful attempts thus far to combat spam have relied on fundamental weaknesses in spam messages or their senders. We review these systems as well as previous network and traffic characterization studies.

Content Filtering: A wealth of content analysis systems are used to great effect today in filtering spam. Learning methods have been effectively applied to building classifiers that determine discriminatory word features [14]. Such content analyzers exploit the fact that a spam message contains statistically different words from a user's normal mail. Even innocuous looking commercial spam, intended to subvert content filters, typically includes a link to an advertised service – thereby providing a basis for differentiation. A popular open source solution is SpamAssassin [8], although there are many competing commercial alternatives. Our system, SpamFlow, does not perform any content analysis on the messages themselves. By providing an alternative classification mechanism, SpamFlow helps address blocking innocuous junk mail, for instance that used to "detrain" Bayesian filters [19].

Collaborative Filtering: Spam is typically sent to many users thereby providing a signature. By aggregating the collective spam of a distributed set of users, collaborative filtering [10, 17] aims to prevent previously marked spam messages from being accepted. For example, popular web

mail clients can easily provide collaborative filtering as their servers are under common administrative control and can leverage spam marked by one user to block spam to other users. Unfortunately, not all mail installations are large enough to take advantage of collaborative filtering and are unwilling to rely on vulnerable centralized repositories. Further, spammers can trivially make each spam unique in an effort to avoid collaborative techniques.

Reputation Systems: Reputation systems attempt to aggregate historical knowledge over specific MTA [7] IP addresses or mail domains. For instance, a large number of messages to unknown recipients might be recognized as a dictionary attack. MTAs that continually send undeliverable mail are given a low reputation. Often, spam honeypots are used in conjunction with reputation systems to gather additional data on spam origination. MTAs which have previously sent spam are likely to continue sending spam. Real-time databases [18, 16, 15] of these offending MTAs and IP addresses provide blacklists which MTAs can query before accepting mail. However, Ramachandran's analysis of DNS blacklists [12] shows that as much as 35% of spam is sent from IP addresses not listed in common blacklists. Their work brings to light an important point about the dynamism of IP addresses in relation to spam. Not only are the IP addresses of botnets changing as hosts acquire new addresses, spammers are rapidly changing addresses in order to evade blacklist reputation schemes. Our SpamFlow, however, has no dependence on IP addresses making it particularly attractive in defending against botnet spam.

Authentication Schemes: Authentication schemes attempt to verify the sender or sender's domain to prevent spoofing-based attacks. Sender Policy Framework [20] limits IP addresses to sourcing mail only for authorized domains. Domain keys [2] uses public keys to associate each email with an entity.

Characterization Studies: Casado et al. perform passive flow analysis on approximately 25,000 spam messages to determine bottleneck bandwidths [3]. Their study finds significant modes at modem, Ethernet and OC-12 speeds, suggesting that spammers employ both farms of low-speed as well as high speed servers. In contrast, we perform a detailed passive flow analysis in order to find relevant features for forming classification decisions.

Brodsky's trinity system identifies botnets by counting email volumes, thereby identifying spam without content analysis. Similarly, the spamHINTS project leverages the sending patterns of spammers to identify the sources of spam. In addition to analyzing server logs, spamHINTS proposes to examine sampled flow data from a network exchange point to obtain a large cross section of email traffic patterns and volumes. For instance, hosts that source email continually or have particular patterns can be identified through a set of heuristics. In contrast, our work analyzes the individual packets of SMTP transactions to obtain much more detailed flow information, e.g. congestion windows and round trip times. Further, SpamFlow relies on machine learning techniques rather than heuristics to build a classification system.

Our work is in a similar spirit to [11] which attempts to characterize the network properties of spammers, for instance the IP blocks to which they belong. Instead, by taking a step down the protocol stack and examining the transport level properties of spam, we hope to take advantage of previously unexploited information.

6. CONCLUSIONS AND FUTURE WORK

Our results are promising, demonstrating that even rough metrics of a flow's character can aid in differentiating incoming emails. We plan to gather a significantly larger data set that includes more valid messages and additional features[3].

Can spammers adapt and avoid a transport-based classification scheme? By utilizing one of the fundamental weaknesses of spammers, their need to send large volumes of spam on bandwidth constrained links, we believe SpamFlow method is difficult for spammers to evade. A spammer might send spam at a lower rate or upgrade their infrastructure in order to remove any congestion effects from appearing in their flows. However, either strategy is likely to impose monetary and time costs on the spammer.

The initial RTT is the strongest indication of spam for our data set. A spammer might attempt to artificially lower the inferred RTT by optimistically acknowledging packets that have not yet been received. However, an adversary cannot reliably know the remote host's initial sequence number for the TCP connection and therefore cannot easily fake the initial RTT. Such attempts to hack TCP to disguise the effects we observe are likely to expose other features, for instance retransmits and duplicate packets.

While RTT is the strongest discriminator on our data, other mail users may have different email interactions with geographically dispersed MTAs. Further, the observed spam RTT may vary for MTAs in countries other than ours. However, such differences demonstrate the strength of a statistical approach. Just as content based filtering is personalized for individual users, the particular features for transport based filtering can be tailored to the end recipients.

Because SpamFlow performs neither content nor reputation analysis, its functionality could be pushed deeper into the core of the network without compromising privacy concerns. SpamFlow is unique in this regard. In addition, with a wider cross-sectional view the performance of SpamFlow would likely improve.

[3]Spam is depressingly easy to obtain, while privacy issues hamper efforts to obtain a large number of ham messages

Utilizing available flow information may aid not only in preventing spam, but also other types of attacks that originate from botnets and compromised machines. For instance, denial of service attacks similarly rely on sending large quantities of data over constrained links. We wish to gather data to better understand the broader applicability of our approach.

Our hope is that this work serves as a step forward in providing a means to combat spam and impose a greater cost on parties sourcing spam.

Acknowledgments

The authors would like to thank Steve Bauer, Arthur Berger and Nishanth Sastry for reviewing early drafts of this work.

7. REFERENCES

[1] M. Afergan and R. Beverly. The state of the email address. *ACM SIGCOMM Computer Communications Review, Measuring the Internet's Vital Statistics*, 35(1):29–36, Jan. 2005.

[2] E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton, and M. Thomas. DomainKeys Identified Mail (DKIM) Signatures. RFC 4871 (Proposed Standard), May 2007.

[3] M. Casado, T. Garfinkel, W. Cui, V. Paxson, and S. Savage. Opportunistic measurement: Extracting insight from spurious traffic. In *Proceedings of the ACM HotNets Workshop*, Nov. 2005.

[4] E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. In *Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet (SRUTI) Workshop*, July 2005.

[5] K. D. Dent. *Postfix: The Definitive Guide*. O'Reilly, 2003.

[6] IronPort. Spammers continue innovation: Ironport study shows image-based spam, hit & run, and increased volumes latest threat to your inbox, June 2006. `http://www.ironport.com/company/ironport_pr_2006-06-28.html`.

[7] J. Klensin. Simple Mail Transfer Protocol. RFC 2821 (Proposed Standard), Apr. 2001.

[8] J. Mason. Filtering spam with spamassassin. In *Proceedings of SAGE-IE*, Oct. 2002.

[9] Messaging Anti-Abuse Working Group. Email metrics report, 2007. `http://www.maawg.org/about/EMR`.

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

[11] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *Proceedings of ACM SIGCOMM*, Sept. 2006.

[12] A. Ramachandran, N. Feamster, and S. Vempala. Filtering spam with behavioral blacklisting. In *Proceedings of ACM Conference on Computer and Communications Security*, Oct. 2007.

[13] S. Rüping. mySVM, 2004. `http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/`.

[14] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. In *AAAI Workshop on Learning for Text Categorization*, July 1998.

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

[16] SORBS. Spam and open-relay blocking system (SORBS), 2007. `http://www.sorbs.net`.

[17] SpamCop. Spamcop, 2007. `http://www.spamcop.net`.

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

[19] J. Vascellaro. Empty spam feasts on in-boxes, Aug. 2006. `http://online.wsj.com/article_email/SB115448102123224125-lMyQjAxMDE2NTA0MjQwODIxWj.html`.

[20] M. Wong and W. Schlitt. Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. RFC 4408 (Experimental), Apr. 2006.

[21] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, 1997.

Appendix: Supporting `tcpdump` traces

I. We observe many instances where our server's TCP socket remains stuck in the LAST_ACK TCP state. After establishing a connection, the remote MTA sends the SMTP HELO and MAIL FROM commands (lines 1 and 4). Our server's TCP acknowledges these packets and responds with SMTP 250 status codes indicating that the command is accepted. Over the next four retransmissions (6-9), our server's response is never acknowledged. Finally, 43 seconds later, the remote MTA sends a FIN (10). Our server sends an ACK (11) but does not shut down the connection because it still has data to transmit. After 20 seconds, the remote MTA sends another FIN (12) and then stops responding.

```
09:51:52.883259 IP 88.245.3.19.2430 > 18.26.0.230.25:  P 1:11(10) ack 31 win 65505
09:51:52.883894 IP 18.26.0.230.25 > 88.245.3.19.2430:  P 31:55(24) ack 11 win 32120
09:51:53.278033 IP 88.245.3.19.2430 > 18.26.0.230.25:  .  ack 55 win 65481
09:52:14.499220 IP 88.245.3.19.2430 > 18.26.0.230.25:  P 11:42(31) ack 55 win 65481
09:52:14.499429 IP 18.26.0.230.25 > 88.245.3.19.2430:  P 55:63(8) ack 42 win 32120
09:52:16.555063 IP 18.26.0.230.25 > 88.245.3.19.2430:  P 55:63(8) ack 42 win 32120
09:52:20.675041 IP 18.26.0.230.25 > 88.245.3.19.2430:  P 55:63(8) ack 42 win 32120
09:52:28.915089 IP 18.26.0.230.25 > 88.245.3.19.2430:  P 55:63(8) ack 42 win 32120
09:52:45.395040 IP 18.26.0.230.25 > 88.245.3.19.2430:  P 55:63(8) ack 42 win 32120
09:52:57.468021 IP 88.245.3.19.2430 > 18.26.0.230.25:  F 42:42(0) ack 55 win 65481
09:52:57.468118 IP 18.26.0.230.25 > 88.245.3.19.2430:  .  ack 43 win 32120
09:53:16.969113 IP 88.245.3.19.2430 > 18.26.0.230.25:  FP 11:42(31) ack 55 win 65481
09:53:18.355043 IP 18.26.0.230.25 > 88.245.3.19.2430:  P 55:63(8) ack 43 win 32120
09:54:24.275059 IP 18.26.0.230.25 > 88.245.3.19.2430:  P 55:63(8) ack 43 win 32120
...
```

II. Often we see a remote MTA connect, acknowledge our server's response but never send any data. After establishing the TCP connection (lines 1-4), our server's MTA identifies itself (5). Without a response, our server retransmits three seconds later (6). The remote MTA then acknowledges the packet (7), but sends no data of its own. Nearly 20 minutes later, our MTA times out and sends a status code 451 SMTP timeout (8) and then a TCP FIN (9). Our server remains in TCP state FIN_WAIT1 as the remote MTA never sends an acknowledgment and has seemingly disappeared.

```
13:40:20.257512 IP 88.234.141.158.48389 > 18.26.0.230.25:  S 776725085:776725085(0) win 24000
13:40:20.257755 IP 18.26.0.230.25 > 88.234.141.158.48389:  S 4032400311:4032400311(0) ack 776725086 win 32696
13:40:23.585053 IP 18.26.0.230.25 > 88.234.141.158.48389:  S 4032400311:4032400311(0) ack 776725086 win 32696
13:40:25.077725 IP 88.234.141.158.48389 > 18.26.0.230.25:  .  ack 1 win 24000
13:40:25.099313 IP 18.26.0.230.25 > 88.234.141.158.48389:  P 1:31(30) ack 1 win 32696
13:40:28.095065 IP 18.26.0.230.25 > 88.234.141.158.48389:  P 1:31(30) ack 1 win 32696
13:40:29.633161 IP 88.234.141.158.48389 > 18.26.0.230.25:  .  ack 31 win 24000
14:00:25.095141 IP 18.26.0.230.25 > 88.234.141.158.48389:  P 31:53(22) ack 1 win 32696
14:00:25.095421 IP 18.26.0.230.25 > 88.234.141.158.48389:  F 53:53(0) ack 1 win 32696
14:00:31.095079 IP 18.26.0.230.25 > 88.234.141.158.48389:  FP 31:53(22) ack 1 win 32696
14:00:43.095043 IP 18.26.0.230.25 > 88.234.141.158.48389:  FP 31:53(22) ack 1 win 32696
14:01:07.095062 IP 18.26.0.230.25 > 88.234.141.158.48389:  FP 31:53(22) ack 1 win 32696
...
```

III. Google MTAs consistently send TCP RST packets. In this example, the Google MTA sends an SMTP quit command (line 1) and then initiates an active close. The active close induces the FIN packet (2) from Google which our server acknowledges (3). Our server passively closes the connection in the other direction by sending a FIN (5). The Google MTA does not acknowledge this FIN and instead sends a RST (6).

```
11:55:57.807504 googl > srv:  P 187089:187095(6) ack 143 win 5720
11:55:57.807510 googl > srv:  F 187095:187095(0) ack 143 win 5720
11:55:57.807628 srv > googl:  .  ack 187096 win 32614
11:55:57.807863 srv > googl:  P 143:167(24) ack 187096 win 32614
11:55:57.808181 srv > googl:  F 167:167(0) ack 187096 win 32614
11:55:57.834759 googl > srv:  R 46149836:46149836(0) win 0
```

IV. Similarly, flows from Yahoo MTAs exhibit TCP resets. This example shows our server responding to an SMTP quit command and then performing an active close on the socket (2). The Yahoo server simultaneously performs a close (3). Our server acknowledges Yahoo's FIN (4). Instead of sending an acknowledgment, the Yahoo MTA sends three resets (5-7).

```
11:20:35.023406 srv > yahoo:  P 113:137(24) ack 1426 win 32120
11:20:35.023782 srv > yahoo:  F 137:137(0) ack 1426 win 32120
11:20:35.023983 yahoo > srv:  F 1426:1426(0) ack 113 win 33304
11:20:35.024073 srv > yahoo:  .  ack 1427 win 32120
11:20:35.076591 yahoo > srv:  R 776208340:776208340(0) win 0
11:20:35.076969 yahoo > srv:  R 776208340:776208340(0) win 0
11:20:35.077381 yahoo > srv:  R 776208341:776208341(0) win 0
```