SMTP Path Analysis

Barry Leiba
IBM Research
Hawthorne, NY
leiba@watson.ibm.com

Joel Ossher
Cornell University
Ithaca, NY
jpo5@cornell.edu

V. T. Rajan
IBM Research
Hawthorne, NY
vtrajan@us.ibm.com

Richard Segal
IBM Research
Hawthorne, NY
rsegal@us.ibm.com

Mark Wegman
IBM Research
Hawthorne, NY
wegman@us.ibm.com

Abstract

Most proponents of domain authentication suggest combining domain authentication with reputation services. This paper presents a new learning algorithm for learning the reputation of email domains and IP addresses based on analyzing the paths used to transmit known spam and known good mail. The result is an effective algorithm providing the reputation information needed to combine with domain authentication to make filtering decisions. This algorithm achieves many of the benefits offered by domain-authentication systems, black-list services, and white-list services provide without any infrastructure costs or rollout requirements.

1 Introduction

Mechanisms to validate the sending domain of an email message are becoming popular, standardized, and hotly debated. The goals of SPF (Lentczner and Wong, 2004; Wong and Schlitt, 2004), Caller-ID (expired proposal), and Sender-ID (Lyon and Wong, 2004; Lyon, 2004) are basically the same: they are each designed to prevent "spoofing" by making it possible for domain owners to publish a list of valid outgoing email servers. Messages that pass one of these tests can be reliably associated with a domain that participated in the delivery of the message[1]. However, this information is not sufficient to filter spam. In addition to knowing a responsible domain, spam filtering requires information about what domains send spam. Most proponents of domain authentication therefore suggest combining domain authentication with reputation services.

This paper presents a new learning algorithm for learning the reputation of email domains and IP addresses based on analyzing the paths used to transmit known spam and known good mail. This information is combined with a novel algorithm for filtering spoofed mail headers to ensure that spammers cannot circumvent the analysis. The result is an effective algorithm providing the reputation information needed to combine with domain authentication to make filtering decisions.

Interestingly, analysis of this algorithm suggests that some or most of the benefits that domain-authentication systems, black-list services, and white-list services provide can be achieved using local learning without the need for extensive adoption of domain-authentications or the costs of third-party black and white list services.

The algorithm we describe uses only the IP addresses mentioned in the standard "received" lines from the headers of an email message (Klensin, 2001) to classify the message as spam or not. It is a learning algorithm, in that we assume the algorithm is trained on a representative set of previously classified mail with the corresponding IP addresses selected. The main intuition behind the algorithm is that mail from the same or similar IP addresses is likely to share the same classification. Experimental evidence suggests that this intuition is true.

This algorithm is very precise at recognizing some spam and non-spam sources, but it cannot accurately label sites for which it has little data. For the rest, a classifier using another technology such as naïve Bayes or Chung-Kwei (Rigoutsos and Huynh, 2004) can distinguish more accurately. For instance, while SMTP Path Analysis is not as accurate as the commonly employed Bayesian spam classifiers, it recognizes information that Bayesian classifiers handle at best generically, and on those parts of that space it does better. Its results can be used to correct erroneous evaluations from a Bayesian classifier, while the Bayesian classifier can classify examples for which

[1] …for some value of "reliably" that is the subject of much debate and controversy. "Plausibly" might be a better characterization, as these techniques are meant to be "best effort" validations.

there is insufficient data for effective path analysis. An aggregate classifier using both results can be better than either.

It is interesting to compare this approach to domain validation schemes such as SPF. SPF lets a domain declare its outgoing e-mail gateways. All mail from that domain "should" pass through those gateways, if the SPF information is correct. If a message passes an SPF check, and we can assume the domain principally does not send spam, then it is safe to pass that mail directly on to a user. But since spammers, too, have registered domains and published SPF records, we cannot assume that mail that passes SPF validation originated from a non-spam domain. There needs to be some means for determining the reputation of those domains.

The algorithm described here uses the IP addresses directly and establishes their reputations, sometimes based on nearby IP addresses, rather than grouping them by an external set of declarations and learning the reputation of the groups. The chief advantages that SPF has in this regard are:

- SPF can group disparate address ranges into a single entity, so less information is needed to create a reputation for that grouping, and
- SPF tells explicitly where the boundaries of the ranges are.

SPF might claim another advantage, in that it can, if the purported sending domain publishes SPF records, distinguish mail that goes through legitimate gateways from mail sent directly from a zombie to the Internet. However, our algorithm is actually good at recognizing legitimate gateways and sorting out mail coming directly from zombie machines (or "botnets"; see Honeynet, 2005), so this advantage is less than it might appear to be. The SPF information could clearly be used in conjunction with our algorithm when available, and when not, the algorithm stands on its own. Note also that, while SPF can't tell anything if the purported sending domain does not publish SPF records, our algorithm can learn from a delivery path regardless of what domain is claimed as the source of the message.

The rest of this paper contains a more complete description of the algorithm, an explanation of the experiments we performed, discussion about those experiments, and our conclusions.

2 Received Line Headers

The SMTP protocol specifies that each SMTP relay used to send an email message must add at the beginning of the message's header list a "received" line that contains (at least) information about the SMTP server receiving the message, from where the server received the message, and a timestamp stating when the header was added. These header lines, taken together, provide a trace of the SMTP path used to deliver a message.

However, the SMTP path listed in a messages received header cannot be fully trusted. The message headers are not signed or authenticated in any way and therefore are easily spoofed. Any SMTP server along the path can insert fake headers that make the message appear to come from any path the sender chooses.

Still, some received line headers are reliable. For instance, all headers that were added by your own domain's inbound SMTP servers can be trusted. A site may also trust the received lines produced by organizations they regularly do business with, assuming they can identify the outbound servers of those organizations. But once the SMTP path implicit in the received lines reaches an unknown or untrustworthy server, the remainder of the purported SMTP path cannot be trusted.

As discussed below, one of the key challenges in developing an effective spam filter based on received-line analysis is determining what portions of the SMTP path recorded in the received lines can be trusted.

3 The Algorithm

SMTP Path Analysis works by learning about the spamminess or goodness of IP addresses by analyzing the past history of e-mail sent using that IP address. The algorithm's learning phase takes as input a set of pre-classified messages that are labeled as spam or non-spam. The learning algorithm extracts from each message the sequence of IP addresses that mail supposedly took to get to the recipient and collects statistics about each IP address. During its classification phase, the algorithm extracts the IP address sequence from the target message and yields a score for that message based on the IP addresses of the gateways supposedly used to deliver the message. The score can be subjected to a threshold to yield a classification of spam or not, or can be used as input to an aggregate classifier. The algorithm looks at no other information; in particular, it does not otherwise analyze the content of the message or consider any domain information.

In the most basic form of our algorithm, the statistics collected for each IP address is simply the number of spam and non-spam e-mails for which it appears. These counts are then used to estimate the probability that mail passing through any previously-seen IP address is spam. The probability estimates are smoothed as necessary to correct for small sample sizes. During classification, we look at the sequence of IP addresses used to deliver the message and assign the message a spamminess score based on the last IP address in the chain for which we have sufficient data.

There are two problems that must be fixed before the above outline of an algorithm is even plausible:

1. Many machines (particularly those at the beginning of the chain, which may be zombies or spammers connecting to their service providers) do not have fixed IP addresses, so the odds of seeing the same IP address in the training set as the one in the message we are trying to classify is lower than we'd like.
2. The above technique is susceptible to spoofing. That is, the message may be coming from a spammy IP address and the machine there may claim that it is passing on a message from a legitimate sender.

We address the dynamic IP issue by combining statistics of the current IP address with those of "nearby" IP addresses whenever there is not sufficient data for the current IP address to make a reliable decision. There are many possible definitions of "nearby" that can be used for this purpose. Our solution is to build a tree of IP addresses that we've seen so far. The root of the tree has up to 256 sub trees, each corresponding to the various possible first bytes of an IP address.[2] Each of those sub trees in turn has up to 256 sub trees itself, each corresponding to the second byte. The same is done for the third and fourth bytes, though, of course, as we go down the tree the branching becomes sparser, yielding a tree with many fewer than 2^{32} nodes.

At each node *n* we store the number of spam messages, S_n and the number of non-spam messages NS_n in which that IP address or range the node represents has appeared. A ratio is computed that is a measure of how spammy the node is, which is $S_n/(S_n+NS_n)$: the number of spam messages divided by the total number of messages that have come through this address or range.

We cannot just use that ratio as it is. Again, there are two issues:

1. What we are trying to record at an interior node is information that will be helpful if we get an IP address with no exact match below that node. That value should be influenced by what happens at the average IP sub range, not what might happen at a few specific IP addresses in those ranges. This may be particularly important in cases where certain addresses are used by spammers but the range as a whole is not, and so we average the activity of the child nodes, not weighted by the quantity of mail that passes through them.
2. If a node has seen only one piece of spam and no non-spam, the odds of the next piece of mail being spam are not 100%.

We solve both problems by the way we actually calculate the score for that IP address. We add an artificial new root with a score of 0.5. We repeatedly go to the subtree that contains the actual IP address if one is available. At that subtree we compute an average of the children of that subtree and the parent. That is, if there are 9 children we take the average of 10 nodes: the parent and the 9 children. For the leaf nodes we take the average of the parent and ratio for the leaf node weighted by the number of messages containing the leaf. Of course, sometimes we don't reach a leaf node if we've never seen this exact IP address in our training set. When we get a new message, we look at each IP address, starting with the last one – the one closest to our receiving machine. We compute its score, a number between 0 and 1, and then combine that with the score for the next IP address. We take a weighted average of the spamminess of the two IP addresses, with weight equal to $1/(s*(1-s))$ where s is the spamminess described above. The rationale is that an IP address that is strongly spammy or strongly non-spammy in the sequence is a better indicator of the nature of the message mail – that the addresses with the most extreme scores are the ones that are most significant to the computation. We continue this process of combining the present average to the spamminess of the next IP address until we reach the end of the list.

As noted above, the above technique is susceptible to spoofing. If a spammer spoofs to foil our algorithm, the mail will appear to come from a legitimate source through a spammy address. To address this problem, we establish a credibility value for each intermediate address, and if an address is not credible we can at least partially ignore the remaining addresses.

After experimenting with the algorithm we found two useful improvements.

We have found that, in practice, if there is any IP address in the sequence that matches exactly an IP address in our training set, it is a better indicator than the score given above when we only find an interior node. So we give more weight to the exact matches.

We have found that there is a distinction between an address that originated messages and one that was a gateway, and we keep separate statistics for originating addresses and intermediate addresses. In particular in our context, when IBM developed its corporate Internet presence, most users in Research Division, who had had Internet email addresses for some time before, moved slowly from gateways inside Research Division to corporate-wide gateways. As spam has increased, the Research gateways now seem to rarely be used for legitimate mail – 98% of what moves through one of

[2] For efficiency, we make the tree sparse, so first-bytes that we have not yet encountered do not appear in the tree. This sparseness continues in all branches of the tree.

those gateways is spam, but some researchers still continue to use it. Hence, mail that goes from there to other parts of IBM would be labeled as probable spam, based on the analysis of the received lines. We fixed this by keeping statistics for the last IP address (the supposedly originating site) separate from all others. So, if an address range **receives** a lot of spam, but all mail **originating** near it is good, then we give it a good score.

4 Experimental Methodology

Our experiments are run against a database that has been collected from an international group of approximately 200 users over many months and contains roughly 170,000 pieces of email. The data was initially labeled by asking the users to vote on all spam and any good mail that made it into their junk mail folder. All 200 users are IBM employees and know that the information will be used for research purposes.

Our data base has been further "cleaned" using a variety of techniques that include clustering of similar messages and hand analysis of outliers. We have been careful to not use the algorithms we are developing or similar techniques in the process of cleaning our database. However, a small number of obviously miss-classified notes arose during our evaluation and have been corrected. The number of such notes is small and does not substantially impact the overall results.

5 Experimental Results

Figure 1 compares the performance of SMTP path analysis to a traditional naïve Bayesian classifier using a standard ROC curve. Each of the algorithms shown in the figure produces a score rather than a black or white decision. The ROC curve shows the different combinations of spam catch rate and false positive rate that can be achieved by selecting different score thresholds for blocking spam.

The SMTP path analysis classifier performs respectably, catching about 70% of all spam with a false positive rate less than one in a thousand. This compares quite favorably to what can be done today with SPF and DNSRBL blacklists. However, its performance falls substantially behind what can be achieved today with naïve-Bayes based anti-spam filters.

What is interesting about SMTP path analysis is that its method of detecting spam is orthogonal to how Bayesian-style text classification works. SMTP path analysis bases its decisions only on how a message is routed, and completely ignores message content. Similarly, the typical naïve-Bayesian classifier cannot make effective use of received lines headers because it knows nothing about how email messages are routed. The result is that combining the two algorithm using classification aggregation techniques can be quite successful.

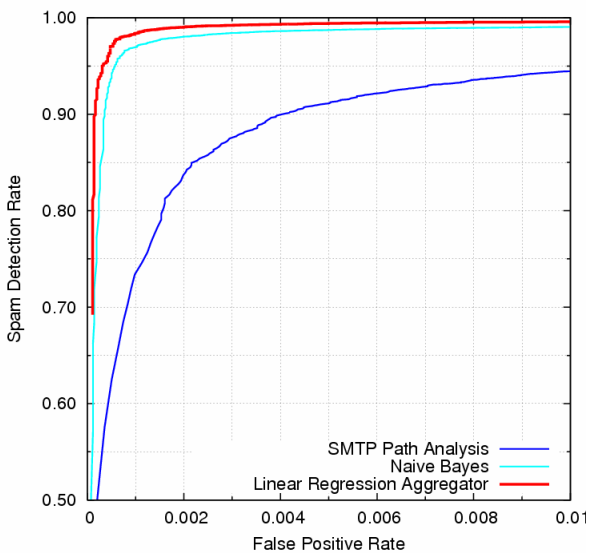

Figure 1: SMTP Path Analysis ROC Curve

Figure 1 also shows the performance of combining naïve-Bayes and SMTP path analysis using a linear regression aggregator (Segal, 2005). The results show that SMTP path analysis can cut the number of missed spam messages in half for any given false positive rate.

Figure 2, on the next page, shows four lines. The red line is for 5,000 trained and 5,000 tested; the dashed blue shows 10,000 each; the green shows 40,000 each; the yellow is for the full DB of 85,000 messages in the test bucket and 85,000 in the training bucket. As can be seen from the graph, the algorithm scales very well. The results suggest approximately a doubling of accuracy of the algorithm with each doubling of the data. The algorithm is also very efficient, since it looks only at a small part of the message. Hence it can be a very useful pre-filter to a more complex algorithm.

6 Discussion and Comparisons

There has been no shortage of standards activities attempting to get a better handle on who has sent email. The most relevant activities are in the domain authentication area, and in improvements to the standards for message-tracking headers. Many of these activities can, once implemented and widely deployed, be combined synergistically with the ideas above.

In our parsing of the received lines we often found missing IP addresses. The current standards

incorporate IP addresses as optional elements in the received line header. If a gateway does not include an IP address we just ignore it and cannot get information for that hop. It might seem, then, that a spammer could just set up a gateway with an implementation that omits the IP address, and the spammer would get away with something. What stops this is the reality that the receiving gateway will put that gateway's address in **its** received line, and it will be picked up there – and we'll learn that it is spammy. Nevertheless, standards that include the IP addresses in a simple-to-parse manner would make our job that much easier.

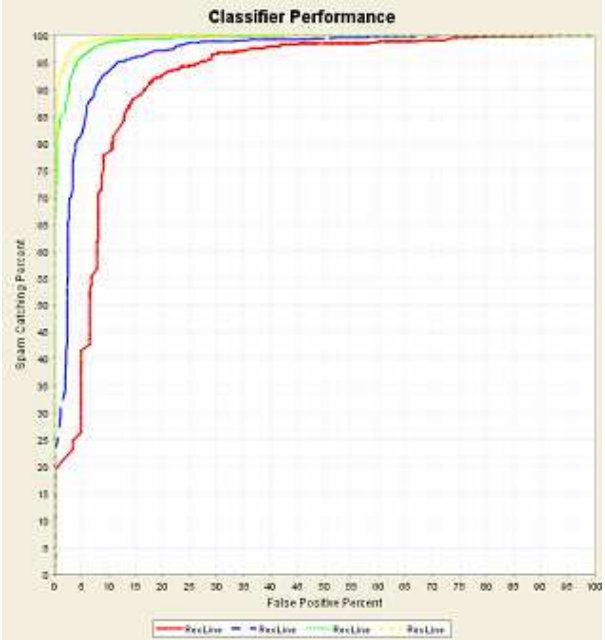

Figure 2: Scaling of SMTP Analysis

Our experimental use of IP address ranges, divided on byte boundaries, has produced very useful results. It is clear, though, that this is not always the right way to determine IP-address relationships. We plan further experimentation with the tree structure, allowing division within bytes of the IP address (to handle a netmask of 255.255.192.0, for example). Cached queries of "whois" databases can also help relate IP addresses that can not be grouped under one netmask. Hosted domains may still be an issue, where two unrelated domains have "nearby" IP addresses by virtue of using the same hosting service. In these cases, though, the hosting service will be the ultimate owner of the address ranges, and must accept some responsibility for the behavior of its customers. We believe that enforcement of terms of service will mitigate this problem; still, more experimentation is needed in this area.

We are looking forward to using the information available from SPF to a greater extent – we have so far done only limited comparisons of our algorithm with SPF, and found ways in which they can complement each other. SPF is becoming widely deployed and we mean to combine it with the above algorithm. Our latest sample of 135k messages, of which about 23k are not spam, shows 15.7k passing SPF tests, with 3k "soft failures" and 2k "hard failures". But spammers have also been registering domains and publishing SPF records, and our Bayesian algorithms find that of the 15.7k passing the SPF tests, 3,584 are spam. We expect, from the known performance of our Bayesian classifier, that at most 4 of those it identified as spam might actually be good mail, so we get the not-surprising result SPF by itself will not block enough spam.

We also note that the fact that we use IP addresses directly, without trying to correlate them to domains or senders (that is, we do not try to validate the sender or detect spoofing, but instead aim to determine the spamminess of the delivery path), avoids the difficulty that SPF has with forwarders and mailing lists. If the path from aol.com to ieee.org to ibm.com is not spammy, it will not matter that ieee.org wound up in the middle of the delivery path. This suggests that our mechanism might be a good complement to SPF.

There are two techniques we intend to try:

- Map all mail from within an SPF domain to a single IP address and then apply our algorithm to the result. We would collapse all addresses within that domain to one entry.
- Insert a unique ID for each SPF domain when mail is sent from anywhere in that domain, at the domain boundary. This would not replace the existing IP addresses, but would add a domain identifier to the sequence.

We have shown that benefit can be derived from examining IP addresses even without using a domain validation mechanism such as SPF. We next discuss the value of the combination of our algorithm with SPF.

For a long time there will be domains that do not deploy SPF, and so the techniques described here can be especially useful for mail coming from them. Moreover, the techniques described here establish a learned reputation system, and may in part be applied to create a reputation service. Many believe, and our experiments agree, that a reputation service is necessary to empower domain validation techniques.

In IBM North America there are about 10 mail gateways, so that 10 times as much data might need to be gathered about IBM. If some of the machines in a domain have become zombies, and the zombies send through the mail gateway, the mail those zombies send will pass the SPF tests. In the algorithm we described

with enough data the reputation for those zombies can be distinguished from that of the rest of the domain, since the zombies used for spam probably send out a lot of mail. The latter cannot be done with a pure domain-based system.

However there are values to a pure domain systems over a pure IP based system beyond needing less data for learning, because a pure IP system can be confused when an organization opens a new gateway in a different part of the IP range from their old gateways. While the organization can make sure the SPF records include the new gateway before it is deployed, it will take some time for our algorithm to learn about it.

Goodman describes mechanisms for and problems with using received lines, since they can't be trusted and can not always be parsed reliably (Goodman, 2004). He specifically develops techniques for determining the boundary between internal SMTP servers which can be trusted and external SMTP servers which may be unreliable. The method presented here nicely sidesteps this issue by learning what IP addresses can be trusted based on past history; thereby, implicitly identifying both internal and external that can be trusted to provide reliable received headers.

7 Conclusions

We have established that examining IP addresses is a valuable addition to the arsenal of tools that the anti-spam community can use. When used in combination with a Bayesian filter it approximately doubles the accuracy that Bayesian filter. Understanding how it works in combination with domain authentication is an important next step both in refining the algorithm and in understanding the value of domain authentication techniques themselves.

Acknowledgements

The authors want to thank the other members of the IBM antispam research team, who have participated in discussions and technical work that have contributed to this paper. Those involved include Nathaniel Borenstein, Jason Crawford, Schlomo Hershkop, and Jeffrey Kephart.

References

Lentczner, M. and Wong, M. "Sender Policy Framework: Authorizing Use of Domains in MAIL FROM", Internet Draft, http://www.ietf.org/internet-drafts/draft-lentczner-spf-00.txt, October, 2004.

Wong, M. and Schlitt, W. "Sender Policy Framework: Authorizing Use of Domains in E-MAIL", Internet Draft, http://www.ietf.org/internet-drafts/draft-schlitt-spf-classic-00.txt, December, 2004.

Lyon, J. and Wong, M. "Sender ID: Authenticating E-Mail", Internet Draft, http://www.ietf.org/internet-drafts/ draft-lyon-senderid-core-00.txt, October, 2004.

Lyon, J. "Purported Responsible Address in E-Mail Messages", Internet Draft, http://www.ietf.org/internet-drafts/ draft-lyon-senderid-pra-00.txt, October, 2004.

Klensin, J. "Simple Mail Transfer Protocol", Internet Engineering Task Force, RFC 2821, April, 2001.

Rigoutsos, I. and Huynh, T. "Chung-Kwei: a Pattern-discovery-based System for the Automatic Identification of Unsolicited E-mail Messages (SPAM)", Conference on Email and Anti-Spam 2004, July, 2004.

Segal, R. "Combining Multiple Classifiers", Virus Bulletin, February 2005.

The Honeynet Project & Research Alliance. "Know Your Enemy: Tracking Botnets", http://honeynet.org/papers/bots/, March, 2005.

Goodman, J. "IP Addresses in Email Clients", Conference on Email and Anti-Spam 2004, July 2004.