Mitigating Email Spam by Statistical Rejection of TCP Connections Using Recent Sender History

Minh Tran, Grenville Armitage
Centre for Advanced Internet Architectures
Swinburne University of Technology
Melbourne, Australia
{mtran,garmitage}@swin.edu.au

***Abstract*- Email spam is a significant problem for ISPs and Internet users. While part of the solution is legislative, there remains many avenues for innovative technological spam-mitigation techniques. We propose a novel TCP-layer algorithm that statistically accepts or rejects in-bound TCP connection requests based on the recent past history of spam injection from particular source IP addresses. Our scheme allows for the automatic rehabilitation of legitimate senders and cuts the operating cost of manually updated blacklists and whitelists. It also reduces the consequences of falsely categorising emails and reduces the last-hop network resource consumption caused by spammers. Our scheme sits transparently in front of the existing SMTP server, so it will supplement (rather than replace) existing spam-filters operating inside existing SMTP servers or at the end-user's mail client.**

***Index Terms*— spam, anti-spam, random rejection, TCP drop/reset, FreeBSD kernel, ipfw, proxy server, mail sever.**

I. INTRODUCTION

Email spam has become a major problem for mail server operators around the world. The solution-space has both legal and technological aspects [1][2], and various schemes continue to be deployed with only modest success. A common approach is to use blacklists and whitelists to identify whether the source of an inbound SMTP connection is to be ignored or trusted respectively. A key challenge is ensuring blacklists and whitelists are kept up-to-date as sites become spammers, are falsely tagged as spammers, or rehabilitate themselves. Traditionally such list updates are performed manually. The use of blacklists and whitelists against unknown senders may be supplemented by content filtering techniques. However, such techniques exhibit finite rates of false-positives (legitimate emails falsely classified as spam) and false-negatives (spam falsely treated as legitimate emails).

Other techniques are economic-based, requiring senders to prove their willingness to send by absorbing a cost on sending traffic. They include Microsoft's 'stamp of approval' method [5], refundable bankable postage [6], e-postage [7] or an improved payment method with differentiated surcharge [8]. Nevertheless, if the spammer meets the 'cost', the mail server will still accept spam emails for delivery. Consequently a high level of network resource, server-side disk and CPU load can be wasted for spam. Moreover, spammers can distribute the computational load across their zombies and reduce the incremental cost to their operation. Thus, 'willingness to incur computational cost to send' is not a strong indicator that the sender is 'legitimate'.

We propose a novel anti-spam technique that should reduce the operational load (on mail server operators) of creating and using blacklists. Our scheme automates the rehabilitation process, thus reducing the human intervention required by mail server and blacklist operators. We also reduce the negative long-term consequences of false-positives [3] and erroneous blacklisting on legitimate email senders.

Our approach operates at the TCP layer and is deployed in front of the SMTP server being protected. We apply a form of statistical rejection of inbound TCP connections based on recent history of spam injection from each sender's IP address. We aim to achieve recommended anti-spam goals [11], allow automatic rehabilitation of legitimate senders who have been previously marked as spammers, and thus reduce operational costs the network and SMTP servers. In the following section we will discuss about the operation and implementation of our scheme.

II. DESCRIPTION OF OUR NOVEL SPAM MITIGATION SCHEME

A. *A high level overview of our anti-spam scheme*

Our new proposal extends a previously published idea to rate limit TCP connections from probable spammers [10]. In [10] we monitored inbound SMTP traffic on a per-source IP basis, and applied severe IP packet rate limiting on any flows that appeared to be carrying spam. We proposed that spam detection occur in real-time inside an SMTP server proxy that would impose any necessary rate limiting. The scheme would 'hurt' spammers by ensuring individual rate-limited SMTP connections took substantial lengths of time to complete. And yet we avoided the negative impact of false-positives by eventually allowing all emails through.

Fred Baker originally proposed the basis of our new scheme in May 2005 [14]. A key limitation of our previous scheme [10] was the consumption of local resources (in the SMTP proxy) to handle packet queues on a per-source IP address basis for every active inbound SMTP connection. Our new approach introduces the notion that an inbound TCP connection may be rejected with some random probability proportional to the level of spam already seen from the connection's originator over some configurable period of time. Rejection is instantiated by returning a TCP RST packet rather than TCP SYN-ACK in response to the inbound TCP-SYN packet. Figure 1 illustrates our model (assuming a prototype

ISBN/ISSN 0977586103

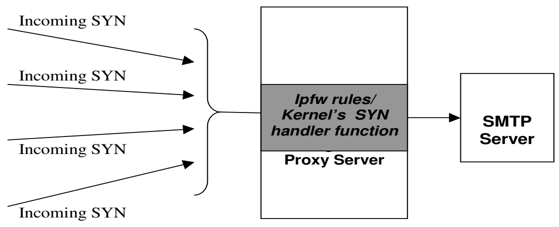

Figure 1 Architectural model – statistical TCP connection rejection implemented as a proxy before a protected SMTP Server

SMTP proxy built using FreeBSD's *ipfw* functionality [12]).

A crucial aspect of our new proposal is that email sources may have their 'reputation' rehabilitated over time without manual intervention. Sources sending acceptable emails should have their TCP reject probability tend towards 0 over time, even if this means rehabilitating an IP source address previously considered to be an unacceptable source. Conversely, sources sending unacceptable emails should see their rejection probability tend towards 1, even if that source had recently been known for sending acceptable emails.

We can describe this scheme by analogy to a concept from active (router) queue management - random early detection (RED), described in RFC2309 [11]. Rather than 'queue size' we measure the changing probability of email being judged 'unacceptable' by a target site's automated spam filters (Figure 2). For each IP address that sends mail to an SMTP target, a moving average between zero and one is maintained, decreasing over time and increasing on each receipt of an unacceptable email. Borrowing directly from RED terminology, we further nominate lower and upper bounds on Q - min_{th} (minimum threshold) and max_{th} (maximum threshold) respectively, such that when Q is < min_{th} all TCP connections are accepted, when max_{th} < Q, all new TCP connections are rejected, and when min_{th} <= Q <= max_{th} new TCP connections are rejected with a probability varying between 0 and maxp (the maximum rejection probability).

External information (from DNS blacklists or local whitelists) is used to permute the parameters Qinit, Qincr, Qdecr, min_{th}, max_{th} and maxp. For example, Qdecr influences the rate at which a source can be rehabilitated. Whitelisted IP addresses might be assigned a faster Qdecr than non-whitelisted addresses. Whitelisted address might also benefit from having a lower maxp than non-whitelisted addresses.

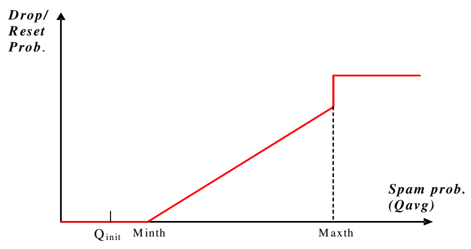

Figure 2 Using RED-like terminology to express connection rejection probability as a function of an IP address' prior spam sending history

max_{th} dictates when the most aggressive TCP connection rejection regime is entered - IP addresses appearing in DNS blacklists could be assigned a lower max_{th} than non-blacklisted IP addresses (rather than blocking them outright).

Our technique leverages the willingness of legitimate mail sources to retry failed SMTP transfers. Over multiple retries (typically separated by tens of minutes) a legitimate email will eventually get through. On the other hand, a spammer either refuses to retry, or wastes a huge amount of time waiting to retry. In the latter cases we have denied the spammer a rapid method of transmitting their spam. By introducing a sliding scale of reject probability we provide a modest level of negative feedback to legitimate sources that are occasionally letting junk or spam emails leak through. (Or who are victim of some unexpected false-positive, such as incorrect listings in a 3rd party blacklist or incorrect classification by the local site's own in-line spam assessment algorithms). At the other extreme, persistent offenders will find their reject probability hovering just below 100%.

B. *An implementation of our anti-spam scheme*

Our prototype has been implemented inside a FreeBSD-based transparent SMTP proxy server. The proxy terminates inbound SMTP connections on behalf of the local SMTP server, and makes TCP connection reject or accept decisions using the algorithm described in the previous section. If a connection is accepted, the proxy establishes a new TCP connection to the local SMTP server and forwards the SMTP exchange line by line. We also implement in-line spam assessment of each SMTP message passing through the proxy, which then permutes the proxy's assessment of a source's willingness to send spam.

Although the random TCP reject functionality could be implemented within the kernel we chose to implement it using TCP packet interception rules in FreeBSD's *ipfw* network layer firewall. Each sender's IP address and their spam-sending behaviour are continuously monitored and dealt with by updating the *ipfw* rule that controls how new SYN packets are treated (Figure 3). Our prototype also reacts to spam coming over an already-accepted connection by randomly closing existing connections if the content-analysis filter detects an over-threshold level of spam from this connection.

C. *Operation of our prototype anti-spam proxy*

At start up our anti-spam proxy server checks a local blacklist file and implements an initial spam history database QList of IP addresses and their Reject probability (Figure 3). Blacklists and whitelists are also used to build entries of TableQ which keeps the list of senders' type (such as blacklisted, whitelisted, or unknown) and their corresponding parameter settings (Qinit, Qincr, Qdecr, min_{th}, max_{th}, maxp).

A list of rules is created inside *ipfw* to identify each IP address from which we wish to reject connections and the associated reject probability. This list of rules is then constantly updated as spam arrives, or sources are re-habilitated.

When a TCP SYN arrives at the proxy, *ipfw* first decides whether to pass the packet up the TCP stack, or to initiate an

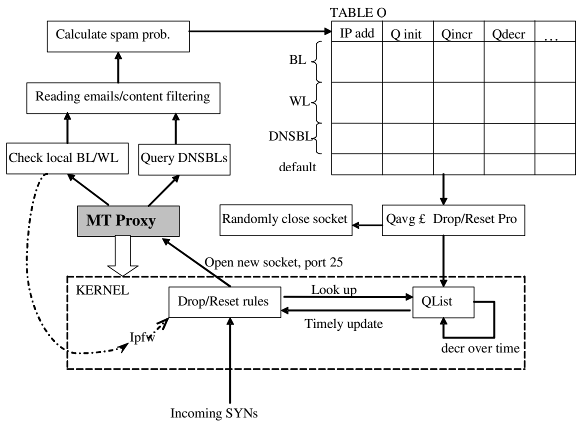

Figure 3 Diagram of the operation of our anti-spam tool

immediate rejection (sending back a TCP RST). If the SYN is rejected, *ipfw* will reject (with 100% probability) any retransmitted SYN packets from the same source for a short, configurable period of time (typically a few tens of seconds, in case the source tries a few quick retransmits to get around a non-zero reject probability). If accepted, *ipfw* allows the two-way TCP session to proceed. New sources have their 'spammer status' queried in DNS black list servers [13] and the resulting information is stored in TableQ and the spam history database QList.

As email flows across the TCP connection the proxy server performs content evaluation. If spam content is detected, the server updates/increments the Reject probability of the sender (at Qincr rate) based on the spam probability and the sender's Qinit. min_{th}, max_{th}, maxp settings. When the sender spam probability reaches a certain threshold, the proxy server also closes (randomly, according to the spam probability) the socket with the sender. The value of Reject probability is also updated in QList.

We keep the following variables for each correspondent IP address in the spam history database:

Address: The IP address of the source.

Q: Ratio of mail received that has been deemed unacceptable by the SMTP receiver's spam filters.

lastReceipt: Timestamp of last update of Q (used to optimise the mathematics of time decrementing Q)

These variables are global to the implementation, whether implemented as compile-time or configuration variables.

min_{th}: The minimum threshold; if $Q < min_{th}$, no TCP SYNs are rejected; otherwise some or all TCP SYNs are rejected.

max_{th}: The maximum threshold; if $max_{th} < Q$, all TCP SYNs are rejected at the rate *maxp*

maxp: The maximum rejection probability

Qincr: The rate at which Q is increases

Qdecr: The rate at which Q is decreases

Qinit: The initial value Q is set to on receipt of an email from a new TCP correspondent.

III. VALIDATION AND ANALYSIS OF OUR ANTI-SPAM SCHEME

A. *Experiment setup and parameter settings*

Figure 4 shows our test-bed with four different types of email senders. We want to evaluate the impact of our schemes on the rehabilitation of legitimate senders and how it will function against spammers.

The spam mitigation proxy server operated at machine *mtran* which forwarded accepted-emails to the mail server at *mtran4*. Each sender used the open source email client *smtpclient* and automated scripts to send consecutive emails at certain intervals. Each email recipient's mail box size was logged on the mail server to. SMTP traffic at mtran was captured with *tcpdump* and post-analysed using *ethereal*. Ipfw was configured to maintain state regarding a previous SYN rejection for one minute (using the *sysctl* variable *net.inet.ip.fw.dyn_syn_lifetime*).

Table 1 shows the parameter settings for each of the sender type in our experiment. Unknown senders start with the initial Reject Probability Qinit as 0 – treated as good sources until experience proves otherwise. Q decrements by Qdecr every minute. Q only increments at Qincr rate if the new content analysis spam is greater than the existing spam probability by at least 5%. Different parameter settings will have different consequences on spammers and legitimate senders (whose emails could be falsely classified as spam or whose machine

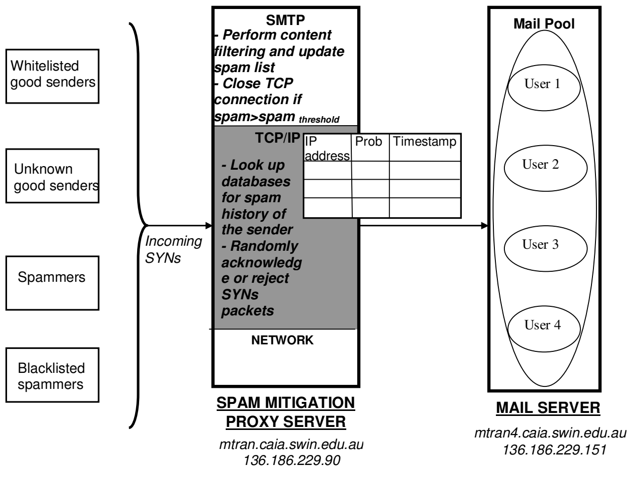

Figure 4 Test-bed Implementation

	Unknown sender	Blacklisted sender	Whitelisted sender
Qinit	0	50	0
Qdecr	5%	1%	10%
Qincr	5%	10%	1%
min_{th}	5	5	5
max_{th}	95	95	95
maxp	95	95	95

Table 1 Parameter settings in the Test-bed

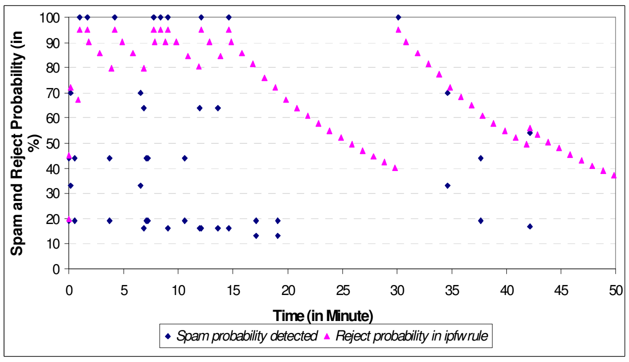

Figure 5 Sender's Reject and Spam Probability as a function of time

could be hijacked and exploited by spammers to send spam emails for a certain period). Evaluating the impact of these parameters will be the subject for future work.

B. *How our scheme functions against spam*

Our first scenario involves artificial 'spammers' sending spam every 10 seconds. They sent spam emails with a mixture of different spam probability level (rated by our content filter) for 15 minutes. Faced with our proxy rejecting their emails with a high probability, the 'spammers' reduced their spam level by sending a mixture of emails with 'good' content (for 5 minutes) and then even took a break (for 10 minutes) before resuming sending us more spam. This time, they sent us spam every 90 seconds (simulating a spammer guessing that the proxy allowed for rehabilitation every 1 minute).

Figure 5 shows spammer behaviour and the response of our anti-spam tool. As soon as the content filter detects spam, the Reject Probability (and *ipfw* rule) is immediately updated. No matter what action a spammer took, their only choice (to increase the chance of email delivery) was to wait for the Reject probability to decrement over time. Sending quickly does not help - many of their emails were rejected (first by the dynamic SYN rule in *ipfw* and second due to their high level of Reject Probability). The gaps on the graph (gaps between blue-colour markers from a period such as 0 to 15 minutes) are periods when spammer emails were rejected.

Figure 6 shows the disk space consumed for a single mailbox over time with the proxy in operation. If the spam

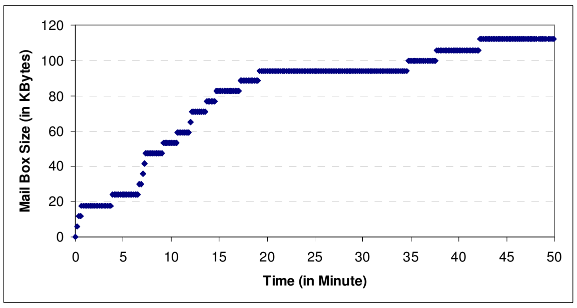

Figure 6 Server Mail Box Size as a function of time

had been arriving unopposed, a linear growth in consumption would have occurred. The non-linear sections in Figure 6 reflect times when the proxy successfully protected the mailbox. Overall, 22 emails arrived in the mailbox (compared to 130 emails transmissions attempted – a saving of 83% in disk space). Our proxy would be similarly beneficial for any SMTP server that stores email before spam-filtering them. In general the network traffic saved over a time period is:

Number of emails/ minute x (Email payload [~5 Kbytes] + SMTP control message exchanges - RST packets)

In our experiment, for the first 20 minutes, the network traffic saved is about:

120 emails x (5 x 1024 Bytes + 1541 Bytes – 54 Bytes)

= 120 emails x 6607 Bytes = 792,840 Bytes = 774.26 KBytes

C. *Legitimate senders and automatic rehabilitation:*

The experiment was done with two types of good senders

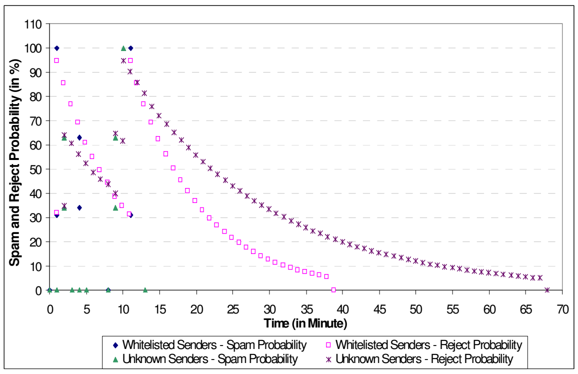

Figure 7 Rehabilitation of Whitelisted and Unknown senders

(whitelisted senders and legitimate unknown senders). Figure shows the rate at which automatic rehabilitation happens.

Legitimate senders who are falsely classified as spam were allowed to rehabilitate. With our anti-spam's current setting (Qdecr = 5% for unknown senders and 10% for whitelisted senders), unknown senders could rehabilitate within an hour while whitelisted senders took less than 30 minutes. (This applies when their emails are rated at 100% spam probability by our content filter. In practice rehabilitation is likely to occur even faster.) In addition, we allow senders to clear their name from the short-term blacklist if their Reject Probability Q reaches 5% and they send no spam during rehabilitation.

D. *Spam IP address distribution and our anti-spam scheme*

Using actual spam traffic arriving at Swinburne on 27th June 2004 we plot a rough estimate of the probability that a particular IP address will send another spam email within the next hour (Figure 8). About 16.5% of spammers sent at least a spam email within a period of 5 minutes from the prior time they had sent spam. This number goes to about 20% for 15 minutes and nearly 30% for 1 hour period.

Figure 9 shows the reaction of our anti-spam scheme to both unknown and blacklisted spammers. If 16.5% or 20% of spammers send another email within 5 or 15 minutes, our proxy is still able to keep the record of their spam history. As a consequence, their emails with be dropped/reset at the rate of 70% and 42% (unknown spammers) or 90% and 82% (black-listed spammers) respectively.

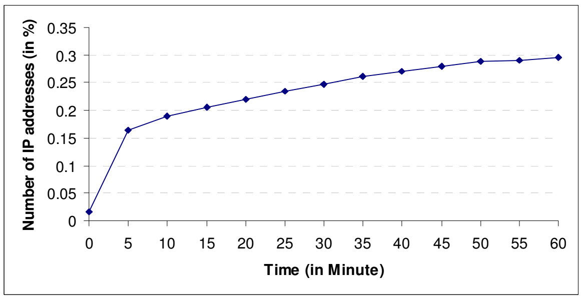

Figure 8 Spammers IP address distribution on 27th Jun 2004

If unknown spammers come back after one hour period and send more spam, the Reject probability will apply immediately as soon as our content filter detects spam. If the spammers delay their action by waiting for one hour for their name to clear, then our scheme has successfully forced them to slow down their spam sending process. Blacklisted spammers are punished even more. In our experiment, after being seen to send an email with 100% spam probability, the connection with the blacklisted spammer was immediately closed and all of the spammer's subsequent emails were rejected. Such a spammer also had to wait much longer for their Reject probability to decrement although it will never go below 50% (default setting for blacklist-Qinit in our scheme).

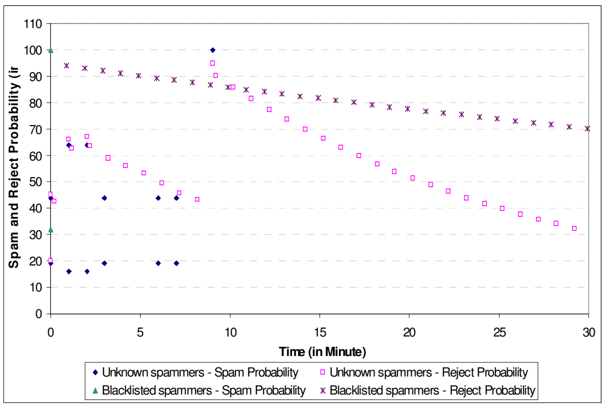

Figure 9 Spam and Reject probability of Unknown and Blacklisted Spammers

IV. Conclusion and Future Work

Most existing anti-spam techniques tend to be built around 'trusted' blacklists that must be manually updated. Such updating requirements lead to increased operator costs and frustration when innocent (or rehabilitated) sites are not removed from blacklists in a timely manner. Other techniques such as content filtering and economic-based ones face the negative consequence of falsely classifying spam sources (either based on the message content or by guessing the source's legitimate intention through their willingness of spending resources on sending emails). Our anti-spam tool differs in that it reduces the costs of false-classification and allows blacklisted IP addresses to be rehabilitated automatically, without operator intervention.

An important extension is to combat spammers who rapidly cycle through closely spaced IP addresses (whether from zombie armies or hijacked/purchased IP address space). We propose to temporarily 'poison' a range of IP addresses either side of an IP address from which spam is seen, proactively raising the reject probability of neighbouring IP addresses that may be about to send spam. Neighbouring addresses that are not subsequently used to send spam will be rehabilitated in relatively short time.

There are many configurable parameters in our anti-spam tool. Future work will evaluate how these parameter settings impact on spam mitigation and the rehabilitation of legitimate senders. Optimal settings will be a function of the spam/legitimate email profile of any given mail server. We are also interested in calculating the sensitivity of our results to the choice of content analysis filter in the proxy itself.

References

[1] S. L. Pfleeger, G. Bloom, "Canning Spam: Proposed Solutions to Unwanted Email", IEEE Security & Privacy Magazine, Vol. 3 (2), pp. 40-47, Mar 2005

[2] House of Representatives, "CAN-SPAM Act of 2003" , http://www.spamlaws.com/federal/108s877.html (as of Aug 2006)

[3] C. Williams, D. Ferris, "The cost of spam false positive", Ferris Analyzer Information Service. Report #385, Aug 2003

[4] L. Zang, J. Zu, T. Yao, "An Evaluation of Statistical Spam Filtering Techniques", ACM Transaction on Asian Language Information Processing, vol. 3 (4), pp. 243-269, Dec 2004

[5] The Penny Black Project, Microsoft Research, http://research.microsoft.com/ (as of Aug 2006)

[6] M. Abadi, A. Birrell, M. Burrows, F. Dabek, and T. Wobber, "Bankable postage for network services", Proc. of the 8th Asian Computing Science Conference, Mumbai, India, Dec 2003

[7] "An Overview of E-Postage", Taughannock Networks, Feb 2004, http://www.taugh.com/epostage.pdf (as of Aug 2006)

[8] Rui Dai and Kang Li. "Shall we stop all unsolicited email messages?" Proc. of the First Conference on Email and Anti-Spam,, Mountain View, CA, USA, July 2004

[9] G. Lindberg, "Anti-Spam Recommendations for SMTP MTAs", RFC2505, The Internet Engineering Task Force, Feb 1999

[10] M.Tran, G.Armitage. "Evaluating The Use of Spam-triggered TCP/IP Rate Control To Protect SMTP Servers," Proc. of Australian Telecom. Networks & Applications Conference, Syney, Australia, Dec 2004

[11] B. Braden and others, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, The Internet Engineering Task Force, Apr 1998

[12] "FreeBSD handbook, chapter 26 Firewalls", http://www.freebsd.org/doc/ (As of Aug 2006)

[13] J. Jung, E. Sit, "An Empirical Study of Spam Traffic and the Use of DNS Black Lists", Proc. of the 4th ACM SIGCOMM Conference on Internet Measurement, Sicily, Itaty, Oct 2004

[14] Fred Baker – Cisco Systems USA, private communications, May 4th 2005