Stopping Spam by Extrusion Detection

Richard Clayton

University of Cambridge, Computer Laboratory, William Gates Building,
15 JJ Thomson Avenue, Cambridge CB3 0FD, United Kingdom

richard.clayton@cl.cam.ac.uk

Abstract. End users are often unaware that their systems have been compromised and are being used to send bulk unsolicited email (spam). We show how automated processing of the email logs recorded on the "smarthost" provided by an ISP for their customer's *outgoing* email can be used to detect this spam. The variability and obfuscation being employed by the spammers to avoid detection at the destination creates distinctive patterns that allow legitimate email traffic to be distinguished from spam at the source. Some relatively simple heuristics result in the detection of low numbers of "false positives" despite tuning to ensure few "false negatives". The system is deployed at a major ISP and has considerably improved the "time-to-fix" for customers who are inadvertently relaying spam and, as a bonus, has proved very effective at detecting a number of recent email virus epidemics.

1 Introduction

Many Internet Service Providers (ISPs) provide email "smarthost" servers for their customers. The customer uses a very simple email client to transfer outgoing email to the smarthost and that machine will then deal with all the complexity of arranging for delivery to remote sites. Some ISPs go further and redirect all outgoing port 25 (the well-known port for the SMTP [5] email protocol) traffic to the smarthost, so as to make use of it compulsory.

In this paper we show how automatic processing of the smarthost's email server logs can be extremely effective in enabling the ISP to detect customer machines that are being used, usually without the customer's knowledge, to send out bulk unsolicited email (spam). Since this log processing is closely related to the monitoring systems that are used for intrusion detection we have dubbed this monitoring of outgoing events "extrusion detection".

There is no specific attribute that is capable of unambiguously characterising outgoing spam – which would permit it to be immediately blocked during the SMTP transfer. Systems for processing incoming email, such as filters, could also examine the content of outgoing email to form a view as to whether it was likely to be spam, but this is expensive, prone to error, and accessing the content gives rise to complex regulatory and contractual issues. However, for a very low cost it is possible to process the summary "traffic data" in the email server logs and form an accurate view, by the application of suitable heuristics, as to whether there is evidence of the sending of spam.

One might expect to detect bulk unsolicited email solely by its volume, but the most effective measure is not its "bulk" nature – which it shares with legitimate traffic such as opt-in mailing lists – but in monitoring failures to deliver (directly related to it being "unsolicited" because the destinations are often invented, out-of-date or refuse to accept the messages). Simple measurements of failure rates can catch other types of legitimate activity, so further processing attempts to distinguish them from the sending of spam. The most effective heuristics have been in detecting the chameleon nature of what is being sent. The senders of spam wish it to be hard to categorise at the remote site and the variability, especially of sender identity, that they use for this purpose provides a very conspicuous indication of the presence of outgoing spam at the sending site.

Clearly, when heuristics are used, there are two possible failure modes. Some abuse may fail to be detected (a "false negative"), and some entirely proper activity may be characterised as spam (a "false positive"). Our experience is that the two main areas of difficulty in reducing both types of failure have been the operation of legitimate, but poorly maintained, mailing lists and the inability of email client programmers to create standards compliant software.

In Section 2 of this paper we set out the background to the spam problem from an ISP's perspective and discuss previous work on automatic detectors. In Section 3 we show what information is available in email server logs and move on in Section 4 to discuss the heuristics that are proving to be useful. In Section 5 we present the results we have achieved and finally in Section 6 we summarise and discuss how extrusion detection may evolve in the future.

2 Background and Related Work

The sending of bulk unsolicited email ("spam") has been a significant problem for Internet users for several years. The senders ("spammers") originally relayed their material through remote servers ("open relays") that would accept email from anyone and then deliver it to its destination. These open relays would accept long lists of destinations for each individual email so that a spammer using a low bandwidth "throw-away" dial-up account could have their message "amplified" by the high bandwidth relay and thereby delivered in bulk.

By 1996 or so, most major ISPs had reconfigured their email servers to prevent relaying except by their own customers. The spammers then switched their traffic to email servers at end-user sites because many of these were still configured to allow relaying, albeit at lower bandwidth levels. The battle against spammers in the later 1990s concentrated on ensuring that end-user email servers were secured against unauthorised relay and in creating support documentation in local languages so that servers outside the English-speaking world could be correctly configured.

As servers became more secure – the major impact having been made by changing system defaults so that they were secure "out of the box" – the spammers changed target again and started to exploit end-user machines directly, even when they were not running an email server. We use the general term "open server" for unauthorised use of end-user machines, with the techniques employed ranging from the exploitation of misconfigured SOCKS proxies through to the deliberate installation of "trojan" software, the latter sometimes being distributed by means of mass mailing viruses. Most recently, since late 2003, the spammers have returned to those customers who are running email servers to utilise default accounts and brute-force weak passwords, so that they can remotely authorise themselves to relay.

Although the main method of sending has changed several times, the reaction at the receiving end has remained constant throughout: hostility and a desire to receive no more junk. Since it is straightforward to inspect the headers and determine where email arrived from, the receivers started to complain. Initially they would write to `postmaster@` addresses to reach the technical staff operating the open servers. However, the end users operating these systems did not always realise the significance of the complaints or even read `postmaster@` email at all. Complaints therefore started to go to the ISP where they began to be dealt with by "abuse teams" operating behind an `abuse@` address, first formally documented in RFC2142 [2].

Meanwhile, the desire to receive no more junk has led to a growth in "blacklists" recording the IP addresses from which spam is sent, so that others can choose not to receive further email from such locations, on the basis that it too is likely to be spam. Besides their obvious rôle in documenting the current set of spam sources, these blacklists have a significant effect in that they put pressure on the sources of spam to fix their problem, because otherwise no-one will accept their genuine email. This pressure becomes intense when an end-user is compromised, spam is sent via the ISP smarthost, and the smarthost itself is blacklisted. Until the listing is removed no customer of that ISP can send email to sites that are using the blacklist. Hence, ISP abuse teams spend a considerable proportion of their time identifying the source of any spam that went via their smarthost, disconnecting customers and educating them on how to correctly configure their software and then, finally, negotiating with blacklist owners to be de-listed. Any scheme that reduces the time-to-detect and the ultimate time-to-fix for comprised customer systems is of direct benefit to an ISP.

2.1 Related Work

The description of abuse team activity given above is one of reacting to incoming reports. Although some proactive work has been done in scanning customer machines for "open servers", the number of possible vulnerabilities makes this increasingly ineffective. In particular, it is generally considered to be unacceptable behaviour for ISPs to attempt to brute-force customer email passwords to check for weakness.

The reactive nature of most anti-spam activity can also be seen in the current emphasis on automatic detection by processing incoming email contents, for which there is a plethora of commercial tools, and automatic reporting of the source, perhaps using the same tools or maybe via a stand-alone website such as SpamCop [6]. Although passing out-going email through a spam filtering system is a possible option for ISPs, the cost of the tools and the need for significant extra computing power make filtering systems expensive to operate and legal issues mean that customer contracts may need to be re-negotiated. Hence, filtering is not currently widely employed on outgoing email.

Email servers generally produce summary statistics (top 50 users, top 50 destinations etc) but these statistics were mainly intended for use in load balancing and no attempt is made to distinguish good traffic from bad. However, these statistics are widely used for abuse detection because the same customers tend to regularly appear on them and therefore "new faces" are likely to be being newly exploited by spammers and will be worth investigating further.

The notion of detecting improper activity in an outgoing direction is clearly related to the analysis of incoming traffic or "intrusion detection", which was introduced by Anderson [1] and Denning [3] and has continued to develop into an active field of research. However, there has been limited interest in investigating outgoing traffic and little work on email handling per se.

Bhattacharyya et al [7] report a Malicious Email Tracking (MET) system which was later generalised by Stolfo et al [8] into an Email Mining Toolkit (EMT). Their approach was to gather detailed statistics on email flows to and from individual accounts and aggregate statistical information on groups of accounts. Abnormal behaviour is then highlighted, whether caused by spam, malicious attachments or virus infections. Since they consider the content of email they drew attention to the privacy issues raised by implementing their system at an ISP. However they do not address the statistical difficulties that arise when several disparate individuals (perhaps in a family or a small business) appear as a single customer from the ISP's perspective.

The EMT system also, by implication, considers legitimate mass mailing to be outside their system whereas the ISP will see this as occasional large flows from an otherwise low-volume user. This problem of "false positives" is less of an issue when only the top 50 customers are being considered. But when data for the whole customer base is processed then false positives can waste considerable time and, perhaps more importantly in practice, lead to disillusionment in the accuracy of detection tools, so that they are ignored even when some of their reports are accurate.

3 Processing Email Logs

We have developed a system to process the email logs generated by an ISP smarthost, so that we can detect any customers that are a source of outgoing spam. The system currently works with Exim [4] logs but very similar information can be found in the logs generated by many other email servers. Since our system condenses log information down into a fairly generic working format, it would be simple to use the same back-end programs and algorithms, but with a different initial processing stage to cope with a different log format as input.

3.1 Email Log Contents

Exim records the arrival of email on the server, noting SMTP protocol level details such as the `HELO` command that is supposed to identify the sending host and the `MAIL FROM` details that are supposed to identify the email's sender. It also records the actual IP address that sent the email and this can be used to determine which customer account was used. Note that the logs do *not* contain any of the content of the email, or potentially helpful spam-detection details such as whether the email contained "attachments" or what the `Subject:` header may have been.

Incoming email is queued for delivery to its destinations (it may be multiply addressed). Each delivery attempt will result in a further logging line indicating whether delivery was successful. After a number of delivery attempts to a particular destination have failed then it is standard practice to pass the email to a secondary "fall-back" system which will continue trying. This prevents the build-up of long queues on the main email server and conveniently ensures that a "complete" record of deliverability is available within a few hours.

We collate the logs every 24 hours and process them through a fairly straightforward Perl [9] program. This creates a single record per email which records date and time, size, who the email was allegedly from, who it was actually from, who it was to be delivered to, and whether it was possible to promptly make a successful delivery.

3.2 Using Delivery Failures to Detect Bulk Email

An obvious characteristic of spam is that it is sent to a large number of addresses. Many of these addresses come from ancient lists and no longer work. Also, many spammers guess at addresses, either working methodically through possibilities in a "dictionary attack" or trying out local parts (left of the `@`) that belong to a user on one particular domain to see if they will reach a real user somewhere else. The net effect is that a key characteristic of spam is that much of it fails to be delivered. For example, this is part of a typical spam run where all of the deliveries failed (needless to say, all the `hinet.net` information was entirely forged):

```
2004-03-01 05:07:25 2jj88@ms32.hinet.net    -> jj88@yahoo.com.tw
2004-03-01 05:07:25 9rwgb@ms29.hinet.net    -> rwgb@sinamail.com
2004-03-01 05:07:26 8sjk642@ms38.hinet.net  -> sjk642@hotmail.com
2004-03-01 05:07:27 6seasons@ms26.hinet.net -> seasons@url.com.tw
```

Therefore one might expect that processing the logs to measure the proportion of outgoing email that fails to be delivered would provide an effective method of spotting outgoing spam – and indeed it does. Unfortunately, when an ISP has a large number of customers then, besides detecting three or four new spam sources per day, it also throws up many hundreds of false positives – which means that further refinement is needed.

4 Avoiding "False Positives" when Detecting Spam

Many systems generate delivery failure reports for the spam that they receive, viz: they accept it and later create a "bounce message". The rejection may be because they are running fancy detectors or because the destination domain is valid, but their system later determines that the particular local part is invalid. These bounce messages should, according to the email standards [5], be sent with a "null" return path, viz: a `MAIL FROM: <>` command should be used in the SMTP protocol. Unfortunately, many end-users are running systems (perhaps home-built) that do not conform to the standards and generate bounces with a non-null return path. For example, these (anonymised) messages are clearly rejections for incoming spam, but the return path is not null:

```
2004-03-01 00:54:33 user@example.com -> NfkGIS@foobarr.com
2004-03-01 01:11:57 user@example.com -> ct4j2L0n7ZJ@samevalues.com
2004-03-01 01:28:36 user@example.com -> ViewCatcher@spicy.ws
2004-03-01 01:48:39 user@example.com -> getyourcard@energizing.ws
2004-03-01 03:47:55 user@example.com -> web.off@ms1.ofmx3.com
2004-03-01 04:30:30 user@example.com -> P39S7RJXldA@eoffersdirect.com
```

Since the source address on incoming spam has often been forged, these rejections will fail to be delivered, generating a rejection report that is returned to the customer, where they probably think the spam load is twice what it actually is! Similar problems arise if "vacation" (or similar) messages are generated for non-existent addresses. The overall effect is that the customer is sending a lot of outgoing email to a large number of addresses that fail – which (as just discussed) is a good heuristic for spotting spam – and hence we have a "false positive".

Some systems set the bounce return path the same as the forward path (perhaps hoping to increase the load at the other end – more likely through ignorance), but the majority use a constant setting for their `MAIL FROM` commands, sometimes a user name (especially for a "vacation" program) but commonly an identity like `mailer-daemon@` followed by their domain. However, spammers believe that their email would be easy to filter out if they used fixed values for `MAIL FROM` so they use constantly changing values, often random addresses at large well-known ISPs such as `aol.com` or `msn.com`. This means that our heuristics can assess failures per sending address and thereby distinguish between spammers and users with rejection systems that are not standards-compliant.

The next problem is that many end-user sites are not the final destination for incoming email. It is common to see email being forwarded (redirected) to members of staff working at remote sites, or to family members who are away from home, perhaps at college or university. These deliveries may then fail, perhaps because the remote site has a spam detector of its own, or perhaps just because the destination mailbox is full. It is conventional to preserve the original `MAIL FROM` when forwarding in this way and this means that the sending site is apparently (indeed actually) a source of spam. For example, some of these emails will be genuine and some will be forwarded spam:

```
2004-03-01 02:45:58 menhongbo@sina.com -> remote.user@example.com
2004-03-01 08:08:18 Dporosity@AOL.COM -> remote.user@example.com
2004-03-01 15:02:44 BA@BritishAirways.com -> remote.user@example.com
2004-03-01 16:18:28 staff@c2m22.com -> remote.user@example.com
2004-03-01 15:54:11 herdoxxvhghhv@yyhmail.com -> remote.user@example.com
2004-03-01 21:06:54 nshffmg@yahoo.com -> remote.user@example.com
2004-03-01 23:52:17 postmaster@thx-trade.com -> remote.user@example.com
```

However, spammers tend to be greedy and send large amounts of email through compromised machines and their email is aimed at multiple destinations. Therefore, it is possible to distinguish these cases: a small number of destinations will be redirected traffic, a large number of destinations is most likely spam and should be investigated.

4.1 Actual Heuristics for Detecting Spam

The intermediate format records derived from the email logs, as described in Section 3.1, are processed on a daily basis by a large and fairly complex Perl program that generates reports about apparent spamming activity. All settings are configurable, the current algorithms and trigger values applied to each individual customer in turn are:

```
Consider emails from each particular source address:
  Destination address identical to source    => assume rejection message
  Failure to deliver > 5, OK delivery <= 100 => assume is a rejection daemon
  Failure to deliver > 5, OK delivery  > 100 => assume is a mailing list
    BUT if >1 mailing list, then do not treat specially
    BUT if >2 rejection daemons, then do not treat specially
Consider destination addresses:
  >4 to same address => forwarded email

Now consider all email that is not a rejection, mailing list or forwarded:
  Report if total score is >100 when adding up the values:
    +10 if receiver rejects with distinctive 'this is spam' message
    +10 if receiver delays acceptance with a 4xx return code
     +1 if receiver responds 'please try later' after RCPT TO
     +3 if >3 destinations for the email, and all fail
  Report if >40 emails where
    1, 2 or 3 destinations and all fail
    or >3 destinations and >25% of them cannot be delivered to
```

4.2 Heuristics for Detecting Mass Mailing Malware

Mass mailing malware (sometimes called "worms" or "viruses") is becoming a significant problem. A rogue program emails copies of itself to remote sites and tries to infect that site by either exploiting a security hole in the email software or by "social engineering" to persuade the user that they should execute the program. In the second half of 2003 a particularly successful program was "Swen" which masquerades as a Microsoft-issued security patch. The malware will locate destination addresses from user address books, the contents of webpages in the browser cache, Usenet news-server meta-data etc. Since these addresses are often out of date, invalid or malformed, the outgoing email will have many delivery failures and it is therefore spotted by our spam detection heuristics.

Malware, naturally, wishes to avoid being detected or trivially filtered when it is received. The practice has therefore taken hold, doubtless copied by one author from another, of obfuscating the `HELO` string – something spammers seldom bother with. It can be difficult to determine the correct setting for the `HELO` string – indeed a great deal of software gets it entirely wrong, our measurements showed 62% of all customers were using a single word `HELO`, whereas the SMTP standard prescribes a fully qualified domain name or an IP address. Malware often avoids all of the difficulty in setting the correct `HELO` string by using random text and, crucially for us, changing that text from one email to the next. For example, with the names anonymised, but the original `HELO` strings:

```
HOST = example.com, HELO = Hhspn
2004-03-01 22:25:56 postmaster@example.com -> user1@findanamateur.com
HOST = example.com, HELO = Zfi
2004-03-01 22:26:42 postmaster@example.com -> user2@uk-personals.net
HOST = example.com, HELO = Xgqdgueg
2004-03-01 22:27:11 postmaster@example.com -> user3@photosound.co.uk
```

Hence, the most effective heuristic is to spot any customer that has employed multiple HELO strings, never re-using the same text. This does yield false positives from companies where multiple employees on different machines each send a single email per day, but as the results in Section 5 illustrate, this is not a major source of "false positives".

In early 2004, after our log processing program had been deployed for some time, the MyDoom program spread very rapidly across the Internet. This malware forges sender domains, and then uses a HELO message that correctly matches that domain. Our system was very effective at detecting MyDoom infected customers (it found just over

1000 in total, 293 on the first day alone). However, it incorrectly identified them as open servers – partly because the heuristics in use at that time failed to give an appropriate score for the presence of "dots" in the HELO messages and partly because the high rate of redistribution meant that many HELOs were used more than once.

The heuristics now being used for malware detection (again, the values are all tunable) are:

```
If more than 10 HELO message are only used once (in 24 hour period)
  then report unless less than the number of HELOs used >1 times
Count how many different HELOs matched the message sender
  report if more than 3 different ones were seen

The report is of a virus UNLESS the majority of HELOs contain dots AND
the average message length < 18K bytes, which we report as an open server
```

4.3 Spotting Email Loops

Having successfully detected customers that were sending spam and those with virus/worm infections, we attempted to validate our results by checking that we were detecting those who were in the Exim statistics reports of the "top 50" users. This showed that we were failing to detect a completely different type of problem, which was email loops.

Loops arise for a number of reasons and different heuristics are needed for each. In the simplest form an email system does not realise that it should be accepting email for a particular domain. It therefore forwards incoming messages to the smarthost, and that returns the email to the true destination, which is the same machine that just failed to accept it. After a few trips around this loop the email will have recorded so many "Received:" headers that the smarthost will give up trying to deliver with a "too many hops" report and record this event in the server logs. This is obviously simple to detect and the customer can be contacted to suggest that they correct their problem.

If the email system at the customer end of the loop is the first to deem the hop count to be exceeded then it will generate the failure. This can be detected by spotting the same message identifier being processed more than once. False positives arise where there is no loop but just a single outgoing message which has already been split into individual emails to each of its multiple destinations – so it is necessary to check if email destinations match as well as their message identifiers.

Some systems manage to avoid increasing the count of Received: headers and so the loop never terminates. This usually arises because some sort of poorly configured "vacation" program is in operation. This may not be spotted by repeated message identifiers, because a new one is generated each time around the loop. However, the destination will remain constant and the size of the email will remain fixed or grow by a small constant value.

Finally, there are loops caused by mishandling message delivery failure reports. Standards compliant email software gives delivery failure reports ("bounce messages") a null return path. This should ensure that if they cannot be delivered, then no further report will be generated. Unfortunately, a poorly written vacation program may respond to the failure report and generate extra traffic. This may not be a problem because the bounces usually come from a "user" such as `mailer-daemon@` that discards all incoming email. However, if two poorly configured systems manage to talk to each other then an email loop, often of significant velocity, will result. We can detect such a loop by using the heuristics already mentioned, but we also treat "speaking to robots", i.e. sending email to one of the dozen or so conventional names like `mailer-daemon@`, as evidence of an impending problem and request that it be proactively dealt with as a customer support issue.

4.4 Leveraging Other People's Detectors

Many systems run complex scanners on their incoming email, trying to detect malware and/or spam. Most systems reject email that is mis-addressed, but some will distinguish between accounts that never existed and those that have been deactivated. This third-party information can be used to make more informed decisions about the type of delivery failure being experienced and hence the nature of what is being sent through the smarthost. Also, although we do not process the content of the email ourselves, we can weigh the evidence of what some remote sites report it to contain. The heuristics in Section 4.1 show how this third-party information contributes to our scoring function.

If the remote site detects a virus then this adds weight to the conclusion we have drawn from the `HELO` values, and the message may give the specific identity of the virus or worm. If the remote site reports that non-existent addresses

are being targeted then this adds weight to a conclusion that this is a spammer who is trying a dictionary attack rather than that the customer is operating a mailing list and failing to remove old addresses.

Recently, some large sites have tried to defend themselves against dictionary attacks by accepting all email. Their view was that when they accepted some email and rejected the rest they disclosed which email addresses were valid. Unfortunately, if an entire spam run is to such sites then all outgoing email is accepted, no errors occur, and the heuristics described here are ineffective. Fortunately, such sites remain the exception at present.

5 Results

We have been running the system described on the smarthost logs of a large British ISP with several hundred thousand customers, operating a mixture of dialup access, ADSL connections and leased lines. The system was originally developed in June 2003 and has been running in continuous "live" service – with daily reports fed to the ISP abuse team – since August 2003.

Initially, there were large numbers of customers being detected as sources of spam, but effective action by the ISP abuse team has reduced the problem significantly and, in particular, reduced almost to zero the number of "repeat offenders" where customers continue to run insecure systems.

For this paper, we have collected detailed statistics for the four-week period starting on the 1st March 2004. There were no public holidays during this period, so the business users were operating five days each week. Many customers send email directly to its destination, but during the measurement period some 84,562 distinct customers used the ISP smarthost, sending a total of 33,393,384 emails to 51,801,043 destinations. (These figures ignore various other usage of the smarthost by internal systems.) The results of running the extrusion detection program on the logs for this period are summarised in Table 1. The false negative figures were obtained by comparing the results with a very much more

Abuse Type	Detected	False +ve	False -ve
Open Servers	56	69	10
Virus Infection	29	6	4
Email loops	14	3	0

Table 1. Effectiveness of Extrusion Detection: 1 March 2004 to 28 March 2004 (28 days)

sensitively tuned version of the program (which generated unacceptably high numbers of "false positives") and also by checking reports from other sources. As can be seen, the open server detection is pretty effective, with a little under half the reports being correct and the others (just 2 or 3 a day) being trivial to reject by manual inspection. The reason for the "false positives" are summarised in Table 2.

Count	Reason
36	Customer operating mailing list(s) with a large number of failing destinations
22	More than 40 genuine delivery failures
4	Customer was forwarding spam to outworkers
4	Varying HELO messages were in fact genuine
2	Misdiagnosed virus infection as spam
1	Misdiagnosed email loop as spam

Table 2. Reasons for "False Positive" Open Server Detection

Obviously, altering some heuristic values would have decreased the false positives, but in each case some genuine spam relaying would have been missed. The "false negatives" were inflated (from 3 to 10) by one particular spammer who, most unusually, relayed email with a fixed sender string to destinations that mainly accepted the email – indistinguishable activity from a legitimate mailing list. They were detected by consulting summaries of active mailing lists and seeing that the same list was apparently being operated by multiple customers.

As can also be seen from Table 1 some excellent results were obtained in detecting customers infected by mass mailing email malware and those who were causing substantial email loops. In the latter case, only those sending 10,000 emails or more in a day are counted, the 867 customers with smaller loops were disregarded.

6 Conclusions

We have shown that a relatively simple email server log processing program can be remarkably effective at detecting ISP customers whose systems are being used for sending bulk unsolicited email (spam). Our system has been able to leverage the attempts by the spammers to make it hard to detect patterns in their activity as a way of distinguishing their material from legitimate email. As a bonus, we have also been able to detect customers who are infected by virus or worm malware and also to spot resource-wasting email loops. The ISP abuse team has been able to promptly act upon the system's reports, often before any other complaints have arrived.

We were fortunate that the ISP we were working with used static IP addresses for all customers including dialup and ADSL users, which simplified our system. Future work will examine methods of detecting "sessions" of continuous activity so that our techniques can be used where intra-day dynamic IP address allocation is in use, without mixing up one customer's traffic with another. Naturally, we will also be aiming to further reduce the false positives whilst continuing to ensure that false negatives remain at a low rate. We are also considering real-time analysis of delivery failures and, where suspicious activity is found, slowing down the rate at which the smarthost accepts email. This will ensure that the spam output is reduced without a catastrophic impact on customer relations when "false positives" occur.

However, one should not become too enthusiastic about this new approach to controlling spam, since it is quite possible that the spammers will adjust their behaviour and adapt to this new threat to their activities. They could restrict their destinations, perhaps sending many different advertisements to one destination rather than the same material to many destinations – this would be hard to distinguish from legitimate email forwarding. They could try and duplicate end-user sender identities and significantly reduce the variation in the identities they use – but this might affect detection rates at the remote sites. They could be less greedy – since lower volumes of transfer would blend in more with other activity – and they might, overall, get extra traffic through before they are detected. Finally, they could disguise their junk as bounces where there is little discernible pattern in genuine traffic – but of course such messages would be trivial to filter at the destination.

We are cautiously optimistic, since we believe that it will continue to be hard to disguise one type of activity as another, so "extrusion detection" will continue to be successful for some time to come. However we do not expect future papers on this topic to contain quite such simple heuristics.

Acknowledgements

We wish to thank Demon Internet for their support in developing this system from its initial experimental concept into a production system, and for their far-sightedness in proposing to release it to other ISPs under the GPL.

References

1. J.P. Anderson: Computer security threat modelling and surveillance. Technical Report, James P Anderson Co., Fort Washington, Pa, 1980.
2. D. Crocker: Mailbox Names for Common Services, Roles and Functions, RFC2142, IETF, May 1997.
3. D.E. Denning: An intrusion detection model. In Proc. IEEE Symposium on Security and Privacy, 1986, pp. 118–131.
4. P. Hazel: The Exim SMTP Mail Server, Official Guide for Release 4, UIT Cambridge, 2003, 620pp, ISBN 0-954-45290-9.
5. J. Klensin (ed): Simple Mail Transfer Protocol, RFC2821, IETF, April 2001.
6. `http://www.spamcop.net`
7. M. Bhattacharyya, S. Hershkop, E. Eskin: MET: An Experimental System for Malicious Email Tracking, in Proc. 2002 Workshop on New Security Paradigms (NSPW-2002). Virginia Beach, Va, ACM Press, 2002, pp. 3–10.
8. S.J. Stolfo, S. Hershkop, K. Wang, O. Nimeskern and C. Hu: A Behavior-based Approach to Securing Email Systems, Proceedings Mathematical Methods, Models and Architectures for Computer Networks Security (MMM-ACNS-2003), LNCS 2776, Springer, 2003, pp. 57–81.
9. L. Wall, T. Christiansen and J. Orwant: Programming Perl, 3rd Edition, O'Reilly, July 2000, 1092pp, ISBN 0-596-00027-8.