

## *Executive Summary*

In response to the growing torrent of unsolicited bulk email, informally known as spam, many technical schemes have been proposed and implemented to distinguish spam from legitimate mail and to block delivery of spam while letting legitimate mail through. Although none of the schemes is the “magic bullet” that some proponents claim, some of them, particularly when used in combination with each other, can help limit the amount of spam that users receive.

## **What do we know about incoming e-mail?**

As Figure 1 shows, mail delivery is a multi-stage process. A mail message originates on a computer known as the *client*, which then *injects* it into the mail system by sending it to the client’s outgoing mail server using SMTP. The injected message includes the sender (From) and recipient (To) addresses of the message, along with the contents of the message. (When sending bulk mail, either legitimate lists or spam, the client often acts as its own outgoing mail server, so there’s no injection step.) The outgoing server then contacts its DNS server to look up the name of the host that receives mail for the recipient domain (known as the MX, for Mail eXchanger.) Once the outgoing server has identified the recipient mail server, it opens a session to that server and uses SMTP to deliver the message. The recipient server may do further processing and possibly relay the message elsewhere, but that depends the way the recipient mail system is set up.

When the recipient server receives an incoming SMTP message, the sequence of events for the recipient is:

- An initial connection request, which includes the numeric IP address of the client machine sending or relaying the mail. If the client’s reverse DNS is set up correctly, the recipient server may also be able to determine a host name for the client.
- Once the connection is set up, the client identifies itself (the HELO parameter) with a name which should be the same as the client’s DNS name.
- The client sends the mail “envelope”, the return address and recipient address(es) for the message (the FROM and TO addresses).

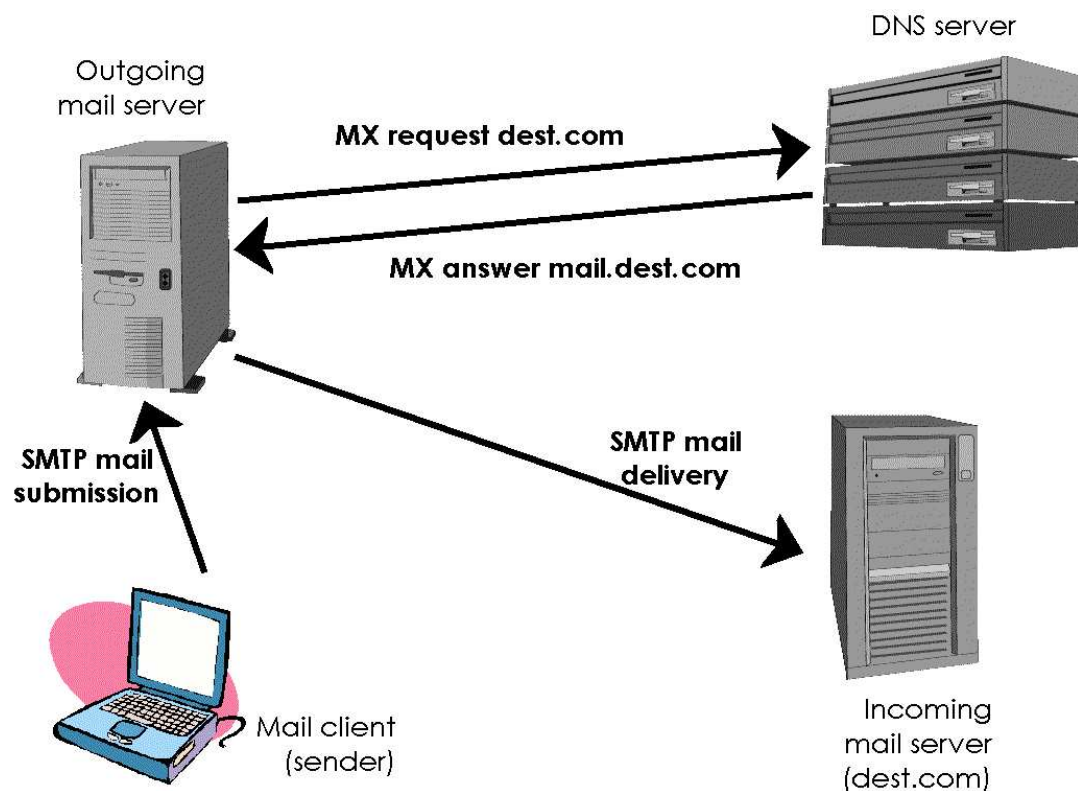


Figure 1: The Mail Delivery Process

- Finally, the the client sends the header and body of the message.

The recipient system can use any and all of these items to categorize mail. It can also collect statistical information, such as the number of messages received per minute by IP address, by sender, or counting messages with similar bodies.

## Source filtering

The most efficient way to filter mail is not to receive it, by blocking mail from undesirable sources. The most common form of source filtering is DNS based blacklists or blocklists, abbreviated DNSBL. A DNSBL publishes a list of IP addresses through the DNS that mail servers can query to decide whether to accept mail from an IP address attempting to make an initial connection. The most common way to use a DNSBL is to reject all mail from addresses listed on a DNSBL, but it's also possible to use DNSBL reports as components in a weighted scoring systems.

The mechanics of setting up a DNSBL are simple enough that anyone with a dedicated network connection and an old PC with Linux can run one. As you might expect, there are now hundreds of public and

private DNSBLs with listing policies ranging from conservative and well-defined to utterly capricious. Few DNSBLs achieve wide acceptance; although the list of popular ones changes every few months, it's rare for there to be more than a dozen that are widely enough used for their targets to notice.

### *Blocklists with mechanical criteria*

The conceptually simplest group of blocklists include IP addresses that meet well-defined and mechanically testable and tested criteria. Variants include lists of open SMTP relays, open proxy servers, hosts that have sent mail to spam trap addresses, hosts that have sent mail via the widely abused "formmail" web script. Listing policies generally involve someone nominating an IP address, testing the address if the criterion requires it (e.g., sending mail to a putative open relay to see if the mail is actually delivered through it), and then adding the address to the list. Some lists remove entries automatically after a while, some retest periodically and remove entries that no longer meet the criterion, some remove on request with or without retesting, a few never remove any entries at all.

Well-run mechanical lists, particularly lists of open relays or proxies, most of which aren't intended to send any mail at all, can block significant amounts of unwanted mail with few false positives (blocking of desired mail.)

### *Shared report blocklists*

Shared report blocklists collect reports of spam (for some definition of spam), and list IP addresses for which the reports pass a threshold. The well-known Spamcop list attempts to track a ratio of spam reports vs. non-spam and lists IP addresses that exceed a predetermined ratio. Since there is little quality control of the spam reports, it's subject to frequent false positives although it identifies new spam sources very quickly.

### *"Opinion" blocklists*

Opinion blocklists are manually maintained lists of addresses that, in the opinion of the maintainer(s), meet a criterion. Some opinion lists include ranges of "untrustworthy" dynamically allocated addresses assigned to dialup users, consumer cable and DSL, that should be sending legitimate mail through their ISP's outgoing mail server, so any mail coming directly from these ranges is likely to be spam. Many ISPs actively cooperate with the major dynamic address lists, and voluntarily provide their dynamic address ranges.

The best known opinion lists, including the original MAPS RBL and the more recent Spamhaus Block List (SBL) and SPEWS lists include IP addresses of known spam sources. The quality of these lists depends entirely on the care and diligence of the maintainers. The SBL, which at this point is probably the most widely used DNSBL in the world, takes care to avoid “collateral damage”, IP addresses numerically adjacent to deliberately listed sources, while SPEWS expands ranges of continuing spam sources for deliberate collateral damage, presumably to increase the pressure on the spammers’ upstream providers. (We say presumably because the SPEWS maintainers are anonymous and don’t always document their decisions.)

### *DNS tricks*

An alternative to DNSBL blocking is to adjust the data available through DNS to make it harder for unwanted mailers to contact recipients by “poisoning” the DNS data that the unwanted mailers use.

Each DNS server that has authoritative data for a domain receives requests from all over the net for that data, including the MX records that identify incoming mail servers. While most DNS servers provide the same data to all requests, it’s not hard to implement “split horizon” DNS that serves one set of data to some requesting IP addresses and other data to other addresses. If a DNS manager doesn’t want mail from certain networks, he can set up split horizon DNS so that requests from those networks for his MX data get “poisoned” values that lead nowhere rather than the actual incoming mail servers. If done well, this is an extremely efficient way to block spam since unlike other schemes, it keeps the spam from being sent to the recipient host in the first place.

A different version of DNS poisoning keeps local users from accessing hosts in undesirable domains. If, for example, a network manager notices offensive spam containing images hosted on particular servers with names in a particular domain, it’s very easy to add poison data to the DNS servers used on the local network that replaces the actual data for the unwanted domains with locally determined data. We’ve used this technique quite successfully to replace some of the more annoying banner ads in mail and web pages with soothing blue boxes hosted on a local web server.

### **Receipt-time filtering**

Once an SMTP server accepts an incoming SMTP connection, it can use a wide variety of techniques to detect and reject spam.

An effective heuristic test is to see if the incoming connection has valid reverse DNS (rDNS), giving the sending host’s domain name as well as IP address. While there’s no technical requirement that all sending

hosts have rDNS, many people have noted that most hosts without rDNS only spam. In mid-2003 AOL started rejecting all mail from hosts without rDNS, which will impel the few legitimate senders without working rDNS to get theirs in order.

Another very effective filter is to validate the domain on the return address of incoming mail by looking it up in the DNS. That is, if an incoming message is from someone@example.com, check that example.com is present in the DNS and has the correct records for mail delivery. This is a cheap way to check for obvious mail forgery, and has few false positives. It used to be a very effective filter, but spammers learned about it and now tend to forge addresses that exist but aren't theirs.

SMTP servers can easily track at this stage the amount of mail sent by incoming IP address, by return e-mail address, or by recipient address, and reject mail that either exceeds bulk thresholds or is sent from known undesirable addresses or to "spamtrap" recipients that get no legitimate mail.

Most spam is sent by "spamware", programs specifically designed to pump out large amounts of mail at high speed, frequently using forged origin data, sending through open relays and other unauthorized intermediaries, and otherwise acting unlike legitimate bulk mail software. Spamware tends to be sloppily written, and its behavior frequently has identifiable technical defects that the recipient host can recognize. For example, when sending the address to which a message is to be delivered, correct software sends in the format `RCPT TO:<user@example.com>`, but spamware often omits the angle brackets or adds extra spaces.

SMTP servers can add deliberate reception delays, since spamware tends to be impatient and give up quickly where legitimate mail software will wait. *Greylisting* is an aggressive but fairly effective technique. It involves rejecting all incoming mail from an unfamiliar IP address with an error code indicating temporary failure. Legitimate mail software will retry in half an hour or so, while spamware usually doesn't bother. Experiments with greylisting suggest that since most legitimate mail comes from familiar IP addresses, the amount of legitimate mail delayed by greylisting is only a few percent.

*Traffic shaping* is another delaying technique with an effect similar to greylisting. The traffic shaping system sits between the Internet and the recipient mail host. Mail that appears legitimate is passed to the mail server in preference to mail that appears to be spam. If the traffic shaping system guesses wrong and delays a legitimate message, the sending host will wait or retry so the message will be delivered eventually, but spamware usually gives up.

## Content filtering

Once the SMTP server has decided to accept a message, the sender transfers the entire set of message headers and the message body. (For SMTP purposes, the message headers are just part of the message, and don't affect message delivery.) Many filtering schemes work on the header and body.

### *Header analysis*

Filters can look for characteristic patterns in message headers generated by spamware. Some spamware puts in obvious mechanical defects like date fields where the time zone name and the time offset from GMT don't match. Other spamware inserts deliberately bogus headers intended to confuse people or software looking for the origin of the spam; these headers often use easy to recognize fixed strings. Spam often uses recognizable From: addresses and subject lines that can be filtered using fixed or variable keyword searches, similar to (or more often in conjunction with) body filters.

### *Body filtering*

The most familiar kind of filter looks for strings in the body of incoming mail messages. It may look words or phrases that appear often in spam, such as Viagra or "Under Bill s.1618" (an anti-spam bill introduced by never passed several years ago) to identify unwanted messages. Body filters often also look for strings that identify desirable mail, such as the return addresses or titles of mailing lists to which local users are known to subscribe. Early body filters used fixed sets of strings, but spammers rapidly learned to avoid them by rewording their messages or using odd spellings such as V\*agra or V.i.a.g.r.a. Current body filters either permit frequent updating of the filter set (as often as several times per hour in the commercial Brightmail system), or use statistical techniques to extract filter rules from sets of mail messages identified as spam and not-spam.

Body filters use essentially the same techniques as virus filters (with different patterns, of course), so if a server is doing body filtering anyway, looking for viruses at the same time adds little extra cost.

### *Bulk counting and shared denouncements*

Since one of the distinctive characteristics of spam is its bulk, a straightforward way to identify it is to count and flag multiple similar messages. The Distributed Checksum Clearinghouse (DCC) service computes hashed checksums for incoming mail, sends the checksums to one of a set of interlinked servers, and gets back a count of previously seen mail with the same checksums. It tracks both full body checksums for exactly identical messages and fuzzy checksums that abstract

out minor differences in different copies. DCC does an excellent job of identifying bulk mail, but it doesn't distinguish between legitimate list mail and spam, so to avoid false positives users have to keep whitelists of the lists to which they subscribe.

Cloudmark (commercial) and Vipul's Razor (freeware) use a slightly different shared denouncement scheme to collaborative filtering. Users send in spam they've received, using hashes conceptually similar to DCC's, which the servers use to compute filtering scores that users can use to see if their incoming mail has been reported as spam. A per-user reputation system attempts to deal with reports from users who send in mail that others don't agree is spam.

All bulk counting systems can be defeated to some extent by "hash busters", random text added to the message to make the hashes of different copies different enough that bulk counters don't recognize them as the same. Early hash busters were strings of nonsense words or text strings at the end of messages; current hashbusters tend to be embedded in HTML messages in ways that don't appear when the message is displayed.

### *"Spammy" behavior*

A variety of body filter looks for patterns that are unlikely to appear in legitimate mail. For example, some spam has large numbers of short HTML comments in an HTML message, which serve only as hashbusters, or words spelled with p.e.r.i.o.d.s between the letters to defeat string filters. Without looking for specific strings, filters can often recognize these unusual patterns that reliably indicate spam.

## Hybrid filtering

While all of the filtering techniques above can be somewhat effective, a combination of many of them usually works better than any individual one.

Many spam filters can be applied in series. Typically a mail server will use DNSBLs to reject some mail, then use body filters on the mail that makes it past the DNSBLs.

The popular hybrid filter Spamassassin computes a spam score for each message based on a user-adjustable (or more often, system manager-adjustable) set of scoring rules that can use all of the schemes described above in its calculation. (Rather than rejecting mail outright on some DNSBLs, it can use the presence of a sender in a DNSBL as part of the score.) A well tuned hybrid system currently does the best job of filtering, but such filters can put a heavy load on the server doing the filtering, and still require frequent updates to the filters and scoring rules as spammers mutate their spam to get past widely used filters.

## Sender identification

Sender identification attempts to recognize mail from known good senders, both to prevent filtering false positives, and to avoid the cost of running known good mail through slow filters. The simplest sender identification is a whitelist of sender addresses, but whitelists are increasingly ineffective both since legitimate senders often change their addresses, and spammers attempt to forge mail from addresses on recipient whitelists.

### *Per user addresses*

Historically most e-mail users have had a single address or a small number of addresses (up to seven at AOL, for example), but there's little technical reason to restrict the number of addresses that people can use. The easiest approach is sub-addresses, so if a user's address is, say, fred@domain.com, any address of the form fred-subaddr@domain.com or fred+subaddr@domain.com is routed to the same mailbox. Users can hand out a separate address to each correspondent (particularly when required to provide an address on a web form), and can thereby tell who incoming mail is from. Since unscrupulous mailers can easily strip off hyphenated sub-addresses, alternatives include moving the actual mailbox into the domain, subaddr@fred.domain.com, or "disposable" random addresses that users can get from a mail service to which they subscribe that are forwarded to their real address. If a disposable address starts getting too much spam, the user can tell the service to turn it off and stop delivering mail to that address.

### *Sender signatures*

For many years it has been possible to sign mail cryptographically, using the same public key cryptography used for sending encoded messages. The two best known signature systems are PGP and S/MIME, which verify that a message came from the e-mail address that claims to have sent it. Although both are widely available (S/MIME is built into the most common mail user programs such as Outlook Express), neither is used by more than a small fraction of mail users. The technology works fine, but getting a key is inconvenient (and, for S/MIME, expensive), collecting a key ring of correspondent keys against which to validate incoming mail is tedious, and signature validation is slow enough that it would be a problem to do for large amounts of mail. Even under the best scenario, there's nothing to keep spammers from signing their mail, so the best that user cryptographic signatures can do is to make one's whitelists and blacklist more reliable.

A somewhat different scheme is for domains, rather than users, to publish a public key, probably via DNS, and use that to sign mail coming from their domain. This prevents spoofing at a domain level, similar to Designated Sender (below.)

### *Sender reputation services*

An interesting alternative to individual signatures is reputation services, in which one presumably trustworthy organization will vouch for the mail sent by others. In effect they are shared whitelists, since recipients generally accept all mail tagged by the reputation services they trust.

The simplest reputation service is Habeas, which licenses a short poem (a Japanese-style haiku) that senders can embed in the headers of their mail, and recipients can easily check using body filters. Enforcement is non-technical, using contract and trademark law to pursue parties that use the haiku without permission or in violations of the terms of the license.

Eprivacy Group's Trusted Email Open Standard (TEOS) is a more complex reputation service that puts a cryptographic signature on mail that contains both a validity check ("this was really sent by so-and-so") and assertions about the nature of the mail ("this is solicited mail related to an existing business relationship.")

### *Challenge/response*

Challenge/response (C/R) systems are an attempt to maintain a whitelist by requiring that senders manually add themselves to each recipient's list. When a recipient mail system gets mail from an unfamiliar address, it automatically sends back a challenge message to which the sender must respond to get the mail delivered. Challenges range from simple "respond to this mail" to complex web based systems in which a user must retype a message from a deliberately distorted image intended to deter automatic systems. Since spammers rarely put a real return address on their mail, this blocks all mail from forged addresses, and even if they do put a real address, the presumption is that no bulk mailer could respond to more than a small number of complex challenges.

Although challenge/response systems are somewhat effective, they have unfortunate side effects. At this point, many C/R systems are so poorly implemented that they don't properly recognize mail from mailing lists and other mechanical sources, and challenge it anyway, annoying list owners and other members of the lists. List owners almost unanimously refuse to respond to C/R challenges, telling list members (with limited success) that it's their job to add the list address to their whitelist.

The obvious way to defeat C/R is to forge the return address of someone already on the recipient's whielist Hence, if C/R becomes popular, we anticipate increasing amounts of spam with forged return addresses "close" to the recipient's as well as disguising the spam itself as a C/R challenge with a URL that clicks through to an advertisement rather than a challenge confirmation page.

### *Designated Sender*

Designated sender (DS) systems attempt to deal with forgery by identifying a set of hosts for each domain from which mail can legitimately be sent. That is, if a message comes from fred@domain.com, the recipient server can look up the sending hosts for domain.com and reject the message if it's not from a listed host. DS schemes could be fairly effective against the significant volume of spam that has return addresses at well-known services like AOL and Yahoo, but isn't sent from those services. (The mail servers of the well-known services all have easy to recognize rDNS, so to get the effect of DS filters for those services, a recipient server can use simple pattern matches on the rDNS host names.) Unfortunately, a small but significant amount of legitimate mail is sent from hosts other than a domain's home network, as when a user is travelling and plugs her laptop into the network of a business or hotel she is visiting ("roaming users"), and DS would reject that mail as well.

Domain signatures, mentioned above, have the potential to address the same issues as DS does. Extended domain signature schemes where domain owners issue subkeys to users to sign their own mail could address the roaming user problem as well.

### **Postage schemes**

Many people have noted that e-mail differs from postal mail in that the recipient rather than the sender bears most of the cost of delivery. If the sender bore the cost, it would deter spam by making it too expensive, as well as possibly redistributing the cost more equitably. Various e-postage schemes have been suggested that would require senders to by electronic stamps that recipients could then cash. Alternatively, "hashcash" schemes require senders to perform slow calculations before recipients will accept the mail, with the CPU time of the calculation serving the role of postage. To date, no workable e-postage scheme has been devised, and as we argue in a companion white paper, *An Overview of E-Postage*, we doubt that one ever will.

### **Conclusion**

The range of technical approaches to distinguishing spam from real mail and to blocking delivery of spam is nearly as wide as the range of spam that it's trying to block. Many schemes are somewhat effective, but spammers can adapt to all of them, creating a continuing technical arms race between spammers and recipients. We believe that technical means can be part of a comprehensive approach to fighting spam, but we don't believe that any technical approaches, individually or in combination, can detect and block enough spam to be deemed a solution to the spam problem.

### Further Reading

Cloudmark, <http://www.cloudmark.com>

ePrivacy Group–Trusted Email Open Standard, <http://www.eprivacy-group.com/teos>

Habeas, <http://www.habeas.com>

Spamassassin, <http://spamassassin.org/>

Spamcop, <http://spamcop.net>

Vipul's Razor, <http://razor.sourceforge.net/>

Revision date: 2004/01/17 06:03:36