ANTHONY HOWE

shoot the messenger

SOME TECHNIQUES FOR SPAM CONTROL

Anthony is a Canadian software developer and sometime system administrator working in the south of France.

achowe@snert.com

EVERYONE HAS THEIR FAVORITE silver bullet for filtering unsolicited bulk email, junk mail, and spam. Unfortunately, those bullets are not perfect and can sometimes end up in your foot.

Each spam control technique has a variety of issues associated with it (as I found after having implemented nine different Sendmail mail filters [http://www.milter.info/], called "milters" in Sendmail-speak). I used them with varying degrees of success for a small ISP in the south of France. I will discuss the techniques I've tried or know about, but the following summary is by no means comprehensive.

SMTP in a Nutshell

The Simple Mail Transfer Protocol (SMTP), RFC 2821, operates based on trust (which is the cause of most of our grief) and cannot be easily replaced with something better for the foreseeable future. The IETF and their Anti-Spam Research Group (ASRG, http://asrg.sp.am/) agree on that much (as their mailing lists reveal after dedicated searching).

Briefly, an SMTP session follows these steps: connection, HELO, MAIL, RCPT, DATA, content, QUIT. Of those seven steps, only the IP address of the client connection and each valid RCPT address specified can be relied upon. Even then, the connecting IP might be questionable; because it's possibly in a dynamic IP address pool, the reverse DNS of the IP is often poorly configured or nonexistent, and now the whois information about IP and domain assignment might be restricted because of privacy concerns (RFC 3912).

As for the other steps, the HELO, MAIL, and message content can be misrepresented or faked. Even QUIT cannot be completely relied upon, since a lot of badly written mail software simply drops the connection when they are done, instead of sending the QUIT command.

Most spam filtering techniques fall into two classes: those that act on the client connection's IP address and envelope information (pre-DATA) and those that act on the message content (post-DATA). The reason I mention this is that once the DATA command is accepted by the receiving server, it is generally committed to reading the entire message until the client indicates it has finished. This, of course, consumes bandwidth and system resources, so some filtering techniques try to make a decision before accepting DATA in order to avoid/reduce more expensive forms of filtering after acceptance.

Challenge/Response

This technique looks at the sender of a message and, if he is unknown to the recipient, accepts and quarantines the message. The server then sends some sort of challenge back to the sender (who must reply, and reply correctly if it's an are-you-human test) before the server allows the quarantined message to be delivered to the recipient. A successful result is typically cached or stored indefinitely.

C/R seems to be the least welcomed of all the possible methods to filter spam. A fair amount of spam and particularly viruses fake the mail address of a real person. So one of two things happens: if the sender is known to the recipient, the message gets through without being caught; if the sender is not known, then odds are the challenge message is sent to a perfect stranger, thus creating even more spam. After a while, this gets to be really annoying for the stranger whose address has been abused. The SpamHaus DNS blacklist considers C/R systems to be just as bad as spam and will blacklist machines using C/R.

DNS Blacklists

Blacklists in one form or another have been used for filtering for a long time, but site-specific lists can be time-consuming to maintain. DNS blacklists make the process simpler by centralizing lists and exploiting DNS caching. The IP addresses of known sources of junk mail are placed on specialized DNS servers. A mail server then queries one or more of those blacklists to see whether the connecting client IP is a known spam source; if so, the connection is dropped or rejected.

Blacklists can be problematic. They first have to receive and identify junk mail or reports of such before they can list the IP address. They must be reliable, responsive, and responsible: you have to count on their information being reasonably accurate, the blacklist service should respond to valid de-list requests almost as fast as they list an IP address, and they must be consistent in their listing and de-listing policy.

Consider an ISP that, while altering the mail server configuration, makes a mistake that goes unnoticed. It is soon discovered to be an open mail relay, which is quickly listed with ORDB (http://www.ordb.org/). The ISP subsequently fixes the mistake and requests to be tested and de-listed as a "spammer." Those requests should be acted upon in a timely, preferably automated, manner. This scenario actually happened at my workplace once, but what was worse was that we consulted ORDB ourselves to reject outside sources, only to find that we had been listed and had started rejecting our own mail! ORDB is generally pretty accurate but is slow to respond to de-list requests and, as a result of a difference in time zones, we were listed as spammers for an entire business day.

Consider what happens now with spam and viruses originating from the dynamic IP pools often used with broadband. One user will have a virus-infected machine (or maybe a "zombie" computer) which sends out a stream of rubbish that results in that IP address being blacklisted. The next day a completely different user connects and is assigned that blacklisted dynamic IP; of course, the new user does not understand why he cannot send any mail. If a DNS blacklist is slow to respond or sets time-to-live values on blacklist entries too long, that IP address can remain blocked for 24 hours or more.

I've tried a variety of DNS blacklists, and the one I recommend is SpamHaus (http://www.spamhaus.org/). ORDB is good too, until it's your machine that's in the blacklist. There are many other blacklists (http://www.sdsc.edu/~jeff/spam/cbc.html), and care must be taken in choosing which to use.

Electronic Postage (Hash Cash)

Hash cash (http://hashcash.org/) is a form of electronic postage by which the sender pays postage in CPU time by performing an intensive computation that is easy for the recipient to validate. A trivial example would be that the sender computes the square root of a large number *Y* and sends the result and the number to the recipient. The recipients can validate the computation by multiplying the given answer with itself to see if it does indeed yield *Y*. Hash cash uses some properties of cryptographic hash functions to achieve the same result. The sender mints a stamp consisting of version number, timestamp, recipient, random data, and how many bits of partial-collision the stamp is claimed to have. Each message contains one hash cash header per recipient.

The idea behind this technique is that the time necessary to compute 1 or 10 hashes for a real person sending mail is insignificant, whereas the time a spammer would require to compute a hash for each recipient, when you consider that they spam thousands or millions of people, would significantly slow down their ability to send junk mail and thus increase their costs.

This is a very nice technique. The drawbacks are its status as a post-DATA verification method, and that both the sender and the receiver have to install software to mint and verify stamps. Therefore, this technique would require wide adoption before it could be used purely on its own to accept/reject messages. However, when combined with other content-filter tools such as SpamAssassin, it can contribute a favorable score toward the acceptance of a message.

Content Filtering

There are several techniques all related to content filtering, where the entire message is received and filtered according to a set of pattern rules, Bayesian statistical analysis, and/or other techniques or external services to verify that a message is good or bad.

One content-filtering technique related to blacklists looks at all the URL domains contained within a message and looks up those domains in a DNS blacklist containing domains appearing in spam messages (http://www.surbl.org/). If a URL in the message contains a blacklisted domain, it's rejected or given a bad score, depending on the filter making the request.

Another technique has the mail server compute a checksum or signature for each message received, then queries it with services that collect signatures for all the spam that their honeypots and users report. Vipul's Razor (http://razor.sourceforge.net/) and DCC (http://www.rhyolite.com/anti-spam/dcc/) are two such services.

Bayesian analysis, as described by Paul Graham's "Plan for Spam" paper (http://www.paulgraham.com/spam.html) and his "Better Bayesian Filtering" (http://www.paulgraham.com/better.html), counts the occurrences of all the words and short sequences found in a large sample of good mail (ham) and an equally large sample of bad mail (spam). The probabilities of each of those words occurring in a spam message are computed. When a new message arrives, the words contained therein are looked up in the probability tables and the top 10 or so of the most significant are used to compute the combined probability that the message is spam or ham.

Bayesian analysis works extremely well once it's trained with very user-specific messages. For example, Mozilla and Thunderbird mail clients use Bayesian identification to delete or redirect mail to a junk folder, and they can be very accurate. But the training has to be continued as spam messages evolve, and "there be the rub." For an individual using a mail client such as Mozilla Thunderbird, it's just a simple matter of toggling an icon on incorrectly classified mail, but if you apply a global Bayesian analysis on an ISP mail server, which is possible, it can require regular maintenance, because the global statistics are not as finely tuned as they would be for an individual.

Another issue with signature and Bayesisn methods is that spammers have reacted by adding benign random words at the top or end of their messages in an effort to throw off the probabilities. Also a lot of spam has gotten shorter in an effort to provide as little content as possible from which to compute a probability and/or to throw off some pattern-based filters.

SpamAssassin (http://www.spamassassin.org/) is one of the notable mail analysis tools. Perl-based, it's a kitchen sink of spam filtering techniques, combining regular expression pattern rules, Bayesian, Razor, DCC, a variety of DNS blacklists (IP, domain, URL), hash cash, and I'm sure some other stuff I've not noticed. Using the combined techniques, it computes a score for each message, which must not exceed a site-specific threshold so as not to be classified as spam. This score can then be used by something like milter-spamc to accept, tag, redirect, reject, or discard a message.

Where I worked, we had a lot of success with SpamAssassin, but it does have its problems—most notably, it's a pig of a Perl process. The process size was about 27MB. I heard of another site that uses SpamAssassin, and their process size was about 107MB. As I recall, SpamAssassin is not threaded, because of Perl; therefore it forks when its message queue gets too long. If you have a large and active group of users (we have about 2,000), SpamAssassin can bring your mail server to a crawl when too many messages arrive in a small interval, such as with a spam or virus attack. This is one of the reasons why many sites use some form of pre-DATA filtering in combination with content filtering, to filter the easy stuff first. I've also heard of another C-based filter called Dspam that can outperform SpamAssassin, so I'm told, though I've not had a chance to look into it yet.

As mentioned above concerning Bayes training, I've learned from personal experience that SpamAssassin can also require a fair amount of hand-holding. The time might be justified if you are defending a large user base, but it appears to require just as much effort for a small number of users. SpamAssassin can retrain/autolearn itself when messages are well above or below the threshold, but when using the Bayes facility sitewide, it's a good idea to configure SpamAssassin to defer rebuilding the Bayes statistical tables on each message to a nightly cronjob, since those tables can be very large (300MB).

Date Conformance and Coherency

The milter-date filter is another very specific form of content filtering I've implemented. A mail message contains several instances of date-and-time information, such as when the message was originally written, possibly when it was resent, and when each mail server en route handled the message. Spam messages often have incorrect timestamps, appear to be too old or too far in the future, and/or demonstrate an inconsistent timeline. The milter-date filter verifies that the date-and-time information within a message is formatted according to RFC 2822, that a message is delivered within a configurable time frame, and that the transit of a message across mail servers reflects a consistent timeline. (Note that SpamAssassin has some date-and-time verification too, though I'm not certain how specific they get.)

One problem with this method is that, surprisingly, too many people have workstations that are set with the wrong time zone, clocks off by a whole year, or similar nonsense. In the two weeks we used this milter, I saw about a dozen or so French users with their Windows workstations set to the Pacific time zone from the day they installed Windows. They just accepted the Microsoft defaults without paying attention. Also, too few servers use Network Time Protocol to keep their clock reasonably accurate, and only in Windows XP has NTP been added as part of the OS. One milter-date user reported that some of Cisco's mail servers were off at one stage and it took two weeks or so to convince a sysadmin of this fact.

Content-Transfer-Encoding Conformance and Coherency

Mail messages have a standard structure and format that is covered by RFC 2822 and enhanced by RFC 2045 and related documents concerning Multipurpose Internet Mail Extensions (MIME). A user's mail software is supposed to adhere to these documents for the formatting and transmission of mail as 7-bit, 8-bit, or binary data. A lot of spam, in particular that written in foreign languages, fails to adhere to these standards, containing unusual, often unprintable, 8-bit character codes in messages that are only supposed to contain 7-bit data for safe and correct transmission between mail servers. Many mail exchanges are very forgiving or careless in what they accept, and so this form of spam gets through. The milter-7bit is another milter I wrote (originally inspired by a mass-mailing worm) to address this class of spam. It ensures that the content of a mail message adheres to the expected or declared Content-Transfer-Encoding as described by the related RFC documents.

This technique is effective, but on its own only catches about 3–5% of spam. The most notable problem with it is that there are many mail-oriented services that fail to correctly specify or encode their mail for transport. For example, a French user might specify her real name with accented letters, but her mail client fails to use MIME word encoding in the From: header to properly specify the name. Also, some legitimate sites such as ebay.com and lhotellerie.fr send email with an explicit Content-Transfer-Encoding: header set to 7-bit, yet include 8-bit values!

Greylisting

Greylisting is a technique that uses the behavior of a normal mail server to delay the acceptance of mail temporarily. When a sending mail server initially contacts a mail exchange to deliver a message, a tuple consisting of an IP, HELO, MAIL, and/or RCPT details is recorded and the mail exchange signals the sending mail server that the message is temporarily rejected. A normal mail server will place temporarily rejected messages into a retry queue and, after an appropriate delay, attempt to resend the message to the mail exchange. The mail exchange, upon seeing the retry from the same tuple as previously recorded, accepts the message. The underlying principle here is that spammers use "mail cannons" to send as much mail as fast as they can and so will not implement a retry queue, as this is too time-consuming when sending millions of messages.

Greylisting is a nice passive technique, and proponents of the technique claim a 90% or better success rate. However, while I've not conducted any statistical analysis on it, from personal observation at my place of work I'd say those success rates are exaggerated or site-specific. And greylisting is not without its problems.

Many of our users work with the money markets and/or they treat email like FTP and instant messaging all in one. They cannot accept or understand that mail might be delayed (just try to explain RFC 2821 limits and delivery timeouts to a French user). Once the first message succeeds, though, the result should be cached for a week or two at the very least, and reset with continued correspondence, else you get an earful of grief on a regular basis.

There are also some very poorly configured mail servers, I'm guessing the pointy-clicky variety, designed and/or administered by people who have no clue about how to get a clue. The four most common issues are: (1) "Hey, I have a whizbang machine with all these CPU cycles to burn, I'll set my queue retry time to 10 seconds"; (2) "If it doesn't get through on the first try, or the second shortly thereafter, I'll wait 12 or 24 hours before retrying"; (3) "I won't retry at all" (some servers with eBay, Amazon, skynet.be, and Southwest Airlines, to name a few: see http://cvs.puremagic.com/viewcvs/greylisting/schema/whitelist_ip.txt); (4) finally, some mail systems are designed to act as a pool—I think gmail.com does this—in that any one of several machines may process the mail queue, and so messages come from different IP addresses.

Greylisting also has the problem of penalizing legitimate mailing list providers until message receivers whitelist the mailing list.

Message Limits

Message limit accounting is a facility to control the number of messages that traverse a mail exchange according to domain, sender, and/or recipient. It could be used on the outbound side, like Hotmail's daily message limits, to limit local users' consumption (particularly if they appear to be infected by a mass-mailing worm); it could be used inbound as an alternative to greylisting; or it could be enabled and disabled as needed during periods of peak mail activity, such as during a virus outbreak or spam holiday season.

I found message limits to be fine for a specific purpose; to be effective as a filtering method, however, they would require more dynamic real-time tracking of which senders are sending from where. For example, I should be able to detect if anthony@example.com attempts to send email from five different IP addresses within the space of 10 minutes, or if the same HELO argument is given for several different connecting clients, or if the same message content arrives from several different IPs.

Callback

A mail exchange that is processing an inbound SMTP transaction looks up, via the domain name system, the mail server responsible for the sender's mail. The mail exchange then opens an SMTP connection back to the sender's mail server and emulates an error return message to the sender without actually completing the transaction (i.e., never issues the DATA command). The mail server being queried normally accepts or rejects the sender's mail address in the early stages of the transaction. The idea here is that spammers use a variety of false and often invalid sender addresses in the SMTP transaction, such as false or nonexistent domains, randomly generated user names from well-known domains, facade mail systems that don't accept any mail, throw-away mailboxes that fill up with errors and replies to unsubscribe, etc.

In order for the callback to work properly, the null address must be accepted (i.e., <>) as required by RFC 2821. Many sites think they are being clever in blocking the null address to avoid spam. Often these sites fail to use more than one filtering technique and look for quick alternatives. Also, many fail to understand why the null address is a requirement in the RFCs. I have successfully educated most sites when this happens, but some others are just too enamored with their lack of prowess to admit they are wrong.

There are also those who completely disagree with this technique. They see it as some form of dictionary attack or an abuse of their mail server's resources (CPU, memory, bandwidth) to have to answer this form of automated C/R (even though the end user will never see a challenge message). One of the arguments against this form of filtering is the claim that spammers are impersonating real email addresses with ever-increasing frequency, so validating the sender's address will have diminishing returns over time and become ineffective.

Call-Ahead

The milter-ahead is a milter that implements a "call-forward" technique, which is similar to a "callback" but intended for use by mail gateways that want to verify, before the gateway accepts the message, that the recipient of a message exists on an authoritative mail store. Think of it as a poor man's LDAP. Many mail systems split the functions of mail transfer and that of storage and retrieval over two or more systems.

Historically, a mail gateway would always blindly accept and forward mail to their mail store, but spammers will often send mail to a domain using a dictionary of user names, resulting in many error message returns, which can sometimes saturate the mail gateway. Often this situation is compounded by the mail gateway queuing those useless error messages for days as they attempt to send them back to spammers who used throw-away domains or mail servers that are now off, eventually resulting in hundreds of double-bounce errors being sent to the mail gateway's postmaster mailbox.

Sequencing Delays

Sendmail 8.13 has a wonderfully simple feature, "greet pause," that catches its fair share of junk mail. When a client connection is established, the SMTP server will send back a welcome message to the client indicating its readiness. The greet-pause feature imposes a site-specified delay before it sends the welcome message. During that time, if the connecting client sends any data across the connection before it has read the delayed welcome response, Sendmail drops the connection. The concept assumes that all well-behaved mail clients must wait until after an EHLO command to determine whether the server supports pipelining. It's assumed that a lot of spam software, in an effort to be quicker and more efficient, pipeline the whole SMTP transaction from the moment the connection is established and never read a response from the server, essentially ignoring any and all errors.

While this method doesn't catch all spam, it catches a decent amount in the earlier stages of the SMTP transaction. I've often wondered why Sendmail hasn't extended the concept a little further to include the other SMTP commands. At the very least it could be applied to the EHLO command; if the mail client uses the older HELO command, then all the SMTP commands could be delayed one or two seconds before returning a response. Slowing down each step of the transaction increases the spammers' costs and reduces their efficiency.

Authorized Mail Sources

There have been several proposals put forward within the ASRG and by independents to specify a means by which a mail exchange can know whether an incoming message comes from a known and authorized source of mail. The idea here is an ISP or business declares the IP addresses of the machines that are responsible for sending outbound mail, then mail from other sources within their IP block can be considered suspect. Solutions like SPF (http://spf.pobox.com/), MTAmark, Yahoo's DomainKeys, and Microsoft's Caller ID (which merged with SPF to create Sender ID) are all variants on a theme.

SPF (classic) and subsequently Sender ID are probably the best known of these proposals. SPF uses specially formatted DNS TXT records to document sources of mail. It's a nice, simple, and elegant solution that any domain owner can manage. However, it has two significant drawbacks. First, all senders must send mail from their domain's SMTP servers, probably using SMTP authentication, which can be tricky to implement or get users to migrate to. But, more important, it breaks any form of relay or mail forwarding where the envelope sender is preserved (not sure if this applies to the Sender ID format). The SPF folks, of course, propose a solution for this: switch from mail forwarding to remailing and use something like the Sender Rewriting Scheme, VERP, VARA, etc., to rewrite the sender address. But there is a catch: these rewriting schemes can significantly increase the length of the user portion of an email address and thus break RFC 2821 maximum limits on the length of the user portion and/or overall address length. Therefore, any filtering techniques that enforce strict conformance to RFC 2821 will see a marked increase in false positives.

MTAmark is similar in nature to SPF but uses reverse DNS instead; it claims it won't break existing mail-forwarding semantics. While I haven't read this Internet draft completely, the one worry I have is that it uses reverse DNS. A domain owner does not have direct control of his IP assignment and must get his IP provider to maintain the in-addr.arpa zone for him. Some might see this as an advantage, by introducing some third-party validation. Also, some IP address assignments are resold several times over, yet the original IP provider may still control the reverse assignment. Therefore, for any legitimate business to modify their reverse DNS, they may have to go up a chain of several levels to get anything done to their in-addr.arpa entries. The other issue with in-addr.arpa is that some IP providers may not pay any attention to the merits of the request, but blindly make the changes.

I know little about DomainKeys other than to say it's patented by Yahoo and involves some form of encrypted signature added to the message headers. This means, of course, that the method is post-DATA and the mail server must accept and read the entire message before it can verify DomainKeys.

It should also be noted that authorized mail source schemes are more directed at "phishing" and "joe job" scams, where the sender of an email message is faked. By knowing the valid sources of mail, you can reject or discredit email. For example, consider a connecting client from aol.com IP space and a sender address of joe@aol.com. With something like SPF you can tell that the IP is not an official source of aol.com mail. These methods can help with spam to a degree, but that was not their original intent. The SPF Web FAQ has an interesting and lengthy section about how SPF can help with spam.

Reputation Filtering

Reputation filtering concerns a mail exchange that can query one or more third-party services for a score based on facts, trends, or reputation of a connecting mail server's IP address and/or the sender's domain. DNS blacklists are a basic form of this, but they provide only simple black/white answers. With reputation filtering, some form of history is gathered concerning the sources of mail and a score or grade is returned, providing more shades of gray.

Meng Weng Wong, of SPF fame, sees reputation and accreditation filtering as being necessary to any authorized mail source scheme, because spammers will publish SPF records, too (many already do). SPF and its derivatives are driving spammers to use their own domains, but they can still jump around the Net or discard their domains at will. But with reputation and accreditation (http://spf.pobox.com/aspen.html), you have a third party that monitors where mail is from and from whom. They can look at objective factors such as longevity, stability, and identifiability. Several of these services already exist, such as Cloud Mark, Outbound Index, and Return Path.

False Positives and Negatives

In my coverage of some of the filtering techniques in use today, I've intentionally said little about false positive (legitimate mail wrongly identified) and false negatives (the failure to identify mail as junk) for the simple reason that I have not collected any statistical data or read any detailed analysis of the techniques. Most of what I've covered here has come from personal experience, study of the techniques, and user feedback related to my milter software.

Whatever methods you end up using, be sure to read up further on the pros and cons, because there has been a lot more said about each of the methods mentioned here than I can possibly convey.

Please Don't Shoot Me

The only thing I can add to all this is, Use more than one method of filtering. Remember, a silver bullet works against werewolves, not against vampires, ghosts, demons, or spam.