# Filtering Spam Email Based on Retry Patterns

Peter Lieven Björn Scheuermann Michael Stini Martin Mauve

Computer Science Institute
Heinrich Heine University, Düsseldorf, Germany
peter.lieven@uni-duesseldorf.de, {scheuermann, stini, mauve}@cs.uni-duesseldorf.de

*Abstract*— **A central problem in today's Internet is unsolicited bulk email: spam. The SMTP protocol lacks a mechanism for verifying the source of a message, and respective protocol extensions are still far from becoming standard. Content-based automatic spam filters are thus often used, and simple origin-based filtering techniques like black- and whitelists are also very common. In this paper we first analyze the retry behavior of Internet mail sources upon delivery errors. We then propose a novel spam filtering approach, founded on the results of this analysis. It is based on the observed reaction of a mail source host to temporary errors. Furthermore, evaluation results from the production use of our filter on real mail servers are given, underlining its very good performance in practice.**

## I. Introduction

Communicating via email has become fundamental to interaction between human beings in business, education and private life. It is easy, convenient, cheap and most people do not want to miss its comfort nowadays. However, the cheapness is also one of the biggest disadvantages in paperless communication, since it led to a problem that many users know well: spam. Many users receive four times more spam than legitimate email [1]. Spam is therefore often considered one of the big problems in current information technology.

The Simple Mail Transfer Protocol (SMTP) [2], the standard for transporting Internet email, dates back to 1982, a time when nobody could foresee in what environment the protocol would be used later. In particular, SMTP lacks a standard mechanism to verify that the sending server and the sender address match. There are approaches like SPF [3] or SenderID [4] to fix this, but as long as no such mechanism is generally accepted as standard, a Mail Exchanger (MX) still needs to accept mail without such authentication.

Most of today's spam filters examine mails after they have been received by the mail exchanger, and classify them. This approach usually performs rather good, but also exhibits some disadvantages. Since the mail has to be received first and complex classification techniques are employed, it is very resource consuming. Futhermore, spammers adapt to cheat the filters and this leads to a never ending competition between the cleverness of spammers and the ability of filters to detect them. The newest development is a heavy increase of so-called image spam—spam that consists only of images, making the content very hard to analyze [5].

But not only the kind of spam, even the way the spam is sent has changed over the last years. Nowadays, the vast majority of spam messages is sent by so-called "zombie machines" or "spambots" [5], [6] . These are compromised end-user systems that act as spam agents. These zombie machines typically use very simple and non-RFC-compliant Mail Transfer Agents (MTAs) for delivering their messages. They are primarily designed for maximum throughput of more or less identical copies of the same message, instead of reliable delivery of regular email to single recipients. These SMTP implementations are often unable to do proper message queueing. In our approach we exploit this: due to their very specific and easily recognizable behavior it is possible to tell the spambots apart from regular, well-behaving mail servers—before they are able to deliver even just one single mail.

According to the SMTP RFC [2], an SMTP server that receives a temporary error when trying to deliver a mail should spool that mail and try again after a reasonable time. The server should also not contact the same destination mail exchanger to deliver some other mail during the backoff period. By denying one or more connection attempts of a mail server with a temporary error and observing its reaction it is possible to tell apart throughput-optimized spambots and regular mail servers with surprisingly high accuracy.

Since all necessary information is collected at connection level our mechanism becomes active before the mail arrives at the MX. The introduced filter forwards a connection only to the real MX if it originates from a well-behaved server acting in a resource-friendly way. This lowers the mail server's load significantly. The thereby saved computational resources can then, e. g., be used to employ more sophisticated content-based filtering techniques in order to detect the spam mail that might have passed through the retry pattern-based filter more reliably.

If the technique proposed here becomes more widespread, it is, of course, possible for spammers to adapt their SMTP engines in order to behave more like regular mail servers. This, however, would also deteriorate the spam mail throughput that can be achieved by the zombie machines significantly. Therefore, even in this case our proposal will help to reduce the total amount of spam mail.

The remainder of this paper is structured as follows. In the following section we review some related work in the area of origin-based spam filtering. Afterwards, in Section III, we present an analysis of the typical retry intervals of spambots versus those of regular mail servers. Based on these results we introduce our retry pattern-based filter in Section IV. Section V is an evaluation of the performance of our filter in production use. Finally, we conclude our paper in Section VI.

## II. Related Work

Spam filtering techniques can be divided into two main groups. First, there are the content-based filters that analyze the headers and the body of the mail and classify it using pattern recognition techniques. This type of spam filters is in some sense complementary to our approach, since we do not look at the mail itself, but only at the behavior of the mail server delivering it. Content-based techniques can then be used to further analyze the mail coming from well-behaving mail servers.

In this section, we will have a closer look at the second kind of spam filters: techniques that can be implemented at the connection level. These do not analyze the mail itself, but consider only the connection delivering it and in particular the origin of the mail delivery attempt, the mail source. After discussing the connection-based techniques, we will also have a look at greylisting [7], a technique that is not purely connection-oriented, but shares some ideas with our own approach.

The most simple way to filter mail is to have static blacklists based on the originating IP. However, in practice it is neither convenient nor viable to blacklist all known spam sources manually. Thus, this method is used only under very rare circumstances. More widely used are whitelists within closed user groups, where one knows exactly whom he wants to accept mail from. Often they are used to prevent mail from certain (e. g., local) sources from being classified as spam by other filters.

A more sophisticated way of blacklists are so called DNS-based Blackhole Lists (DNSBLs) or Realtime Blackhole Lists (RBLs). They are generated by a provider based on the observation of mail sources. A host is added when it has been identified as a spam source. The list is then distributed via the Domain Name System (DNS) [8]. DNSBLs are widespread nowadays and can be very efficient. However, it takes some time until a spam source appears in the blacklist, in our experience usually 2–4 hours. Note that this means that DNSBLs force spammers to send their mails in a very short timeframe—often only in a “single shot” [6]—before they are listed and locked out. Our approach therefore ideally complements DNS blackhole lists: it delays mail delivery especially for aggressive mail sources, and thereby often allows for spammers being blacklisted before mail would be accepted.

A common way of filling whitelists automatically is the use of Challenge and Response Systems (CRS), like introduced in the Tagged Message Delivery Agent (TMDA) [9]. In a CR system every yet unknown combination of sender address and IP is initially considered harmful. The corresponding mail is spooled and the sender gets a challenge sent back via email. Usually the challenge is to click a link or to enter a code. Once the system gets a valid response the mail address/IP-combination is whitelisted, the corresponding mail is released and all further mail will be immediately delivered. Almost no spam makes it through such a system. However, additional measures are necessary to prevent the local server’s mail queue from clogging with undeliverable challenge mails to non-existent, faked spam mail source addresses.

Spammers often tend to so-called hammering, i. e., connecting repeatedly with little or no time between the connection attempts. It has been proposed to lock out hosts hammering the local system for a certain amount of time or to throttle the allowed maximum bandwidth for resource wasting IPs [10]. One effect of our approach is just that: very aggressive hosts may not connect.

A very interesting approach is detecting spammers by malicious activities: an Intrusion Detection System (IDS) can be coupled with the spam filter [11]. This relies on the fact that many of the sending hosts are compromised end-user systems that try to infect the MX itself before starting to send spam to it. If the IDS detects malicious behavior of a host, this is a strong hint that it is not a legitimate mail server. We have adopted this idea as an optional extension to improve our system further.

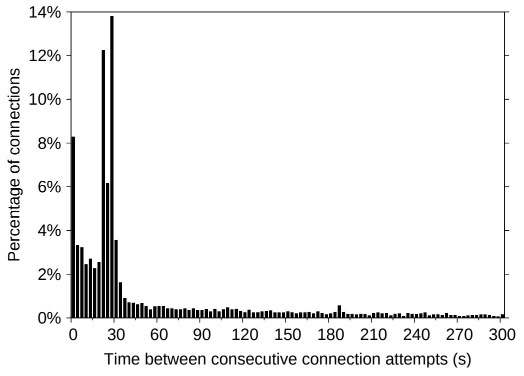
One of the latest developments in spam filtering is greylisting [7]. Greylisting looks at a triple of sender mail address, source IP, and destination mail address. For a fixed amount of time starting from its first occurrence, all connections using this triple are rejected with a temporary error. After this fixed time has elapsed each mail with that triple is accepted. According to the SMTP RFC the sending server should spool that mail and try again after a reasonable amount of time.

Greylisting works under the assumption that spammers often do no queueing or use a different sender address when trying the next time. Greylisting performs very well at the moment and is able to filter up to 90 percent of incoming spam connections while generating nearly zero false positives [12]. However, greylisting uses static timers and does not care about the senders’ behavior. For greylisting, it does not make any difference whether the mail source connects once or one million times during the greylisting period. Thus, it loses valuable information that can be exploited to make decisions faster and more accurate. Furthermore, greylisting considers mail address pairs, whereas we concentrate on the IP address of the mail source. We consider this more appropriate, since a mail source will typically either be a spambot, or a regular, well behaved mail relay. Therefore, we classify source hosts rather than communicating users. In some sense, our approach could be considered as a mail source-centric, much more adaptive form of greylisting.

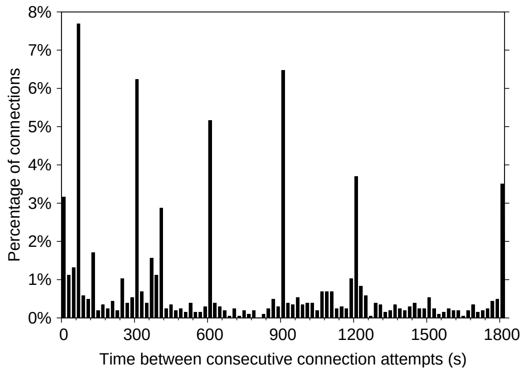
## III. Analysis of MX Retry Behavior

After a connection attempt has been rejected with a temporary error, mail servers wait for some time until they reconnect. In this section, we analyze the retry behavior of real mail hosts on the Internet, and we show that the retry pattern of spambots is considerably different from that of regular mail servers.

All data collection and testing was done on the mail server of dlh.net [13]. This server contains about 100 mailboxes, most of them existing for almost ten years. These long-existing mail addresses, that were often carelessly posted in newsgroups or appeared in plain-text on webpages, attract a lot of spam. The



(a) Bad mail sources (interval size: 3 s).



(b) Good mail sources (interval size: 20 s).

Fig. 1. Distribution of retry times for good and bad hosts.

data collection phase ran over four weeks between May 5th and June 4th 2006.

In our test setup we refused each connection from a yet unknown IP for the first two hours after its initial connection attempt. During these two hours we recorded the time between successive connection attempts from the same IP address. Afterwards every incoming connection was permitted. The main problem that arises is to determine which of the hosts are “good”, and which are the “bad” ones. In order to identify these two groups in our collected data, we used additional, independent information collected on the connecting IPs.

The most important criteria to identify a spambot were DNS-based blackhole lists, and the DNS PTR records of the spam sources. All IP addresses that were blacklisted by common DNSBLs are considered to be bad. In the specific situation of our sample data collection, the delay until some IP is blacklisted is not a problem: if some IP for which the retry pattern has been observed was blacklisted later, we know that the pattern belongs to a spam source. Via their PTR record, many dialup hosts can be identified. These are also most likely zombies. If a host has no PTR record, or if the authoritative DNS server is not responding, this is also a strongly negative indication.

On the other hand, if a mail passes the challenge-response system installed on our server, it is extremely likely that it is not spam. Thus, we consider a mail source to be good if some mail sent by it successfully passed the CR system.

The primary mail exchanger for dlh.net is mx.dlh.net. In order to identify even more bad hosts, we set up a fake secondary MX with lower priority which always denied incoming connections. Connection attempts to this host were logged. According to the SMTP RFC MXs have to be contacted in the order of their priority. Spambots often do not respect this. Many implementations even seem to ignore DNS MX information completely, and connect to mail.xyz.foo for mail to the domain xyz.foo, without performing a real MX lookup. So, we set up another fake server named mail.dlh.net. Our records revealed that every single host that contacted mx2.dlh.net or mail.dlh.net instead of mx.dlh.net first did not pass the challenge-response system later on. This means that there are no “good” mail servers which use the DNS MX information incorrectly. Therefore, we consider all those hosts bad as well.

Finally, we set up a third logging mechanism logging all connection attempts to any ports except TCP port 25 (SMTP) and port 179 (ident). These logging facilities reveal information about whether a mail source tries to scan or even compromise our system before sending mail to it. Both are strong signs for the mail source being a spambot. This idea was partly inspired by [11].

Using the mentioned criteria we are able to identify good and bad hosts in the recorded retry data. 39 000 mail sources were identified as bad, 13 800 of these connected more than once and could therefore be used for our analysis. 1 300 hosts were classified as good. For the rest, neither our positive nor the negative criteria apply. This was the case for 100 100 hosts, out of which 23 000 did at least one reconnect.

Figure 1 shows the distribution of retry times for the two groups of mail sources. For the bad ones, there is an accumulation of very short retry times below 30 seconds, many are even below one second. There are strong indications that most unclassified hosts are also bad: good mail hosts will typically get their mail through the CR system eventually, thus good hosts are recognized with high probability. Furthermore, the retry behavior of the unclassified hosts was typically very aggressive, similar to that of the bad hosts in Figure 1(a).

Good senders on the other hand will likely reconnect after fixed intervals forming clusters at characteristic times like 60, 600 or 900 seconds, as can be seen in Figure 1(b). However, even a small fraction of the good senders seem to have broken or unfriendly SMTP error code handling and reconnect quickly after a temporary error. But as it turned out, these servers reconnect only once after a short backoff, and do appropriate queueing afterwards. Our approach is well able to deal with this, and it is thus not a relevant problem in practice.

## IV. Retry-Based Filtering

As it has been shown in the previous section, the typical behavior of a spam source and that of a regular mail server differ very significantly. This is exploited by the spam filtering mechanism proposed here. In our implementation, the filter is realized in an SMTP proxy. This special proxy observes and evaluates the retry behavior of each source IP. For each incoming connection attempt it then decides whether to reject it with a temporary error, or to forward it to the real mail exchanger.

Our filtering technique is based on a concept which we call the *observation period*. Each mail source that attempts to establish a connection has its own observation period, which begins at the initial connection attempt. During the observation period, all connection attempts of the mail source are rejected and its retry behavior is recorded. When it connects after the observation period has elapsed, the connection is allowed. The length of the observation period, however, is not fixed. Instead, it is dynamically adjusted depending on the observed behavior. Initially, it is set to 15 minutes. If bad behavior occurs during the observation period, it is prolonged. This has two implications. On the one hand, if a host stands out negatively, it will be observed for a longer time, which allows for the collection of more data on its behavior. On the other hand, the longer observation period increases the probability that a spambot is identified by, e. g., a blacklist provider, and listed on a DNSBL before it would otherwise have been let through.

A possible prolongation of the observation period (we call this a *penalty*) depends on the kind of bad behavior that has been observed. Extremely bad behavior—like, for example, hammering or an attempt to compromise the mail host—leads to a long penalty, while "moderately bad" attitude just prolongs the observation period slightly. In the following, we will look at our implementation in a little more detail. The specific times that we use in our approach are chosen heuristically, based on experience and an in-depth analysis of the data set that has been described in the previous section.

The most central value in our filter is the expected retry time $t_{\text{ER}}$. If a reconnection attempt is made before the expected retry time is over, this is punished by prolongating the observation period. The SMTP RFC proposes a value between 15 and 20 minutes between consecutive connection attempts. As our previously described experiments show, such a long period is unrealistic even for many friendly mail servers. We use an expected retry time of 180 s in our implementation. Our experiments show that with this value, well-behaving mail servers are let through with low delay, while on the other hand spambots are locked out quite effectively. The penalty $P$ of the observation period is based on the difference between $t_{\text{ER}}$ and the observed retry interval $\Delta t$. Furthermore, consecutive short retries are accounted for. We calculate the increment of the observation period for the $n$-th consecutive early reconnection attempt as

$$P = (t_{\text{ER}} - \Delta t) \cdot n. \qquad (1)$$

TABLE I

Penalties for Observed Malicious Mail Source Behavior.

| Observation | Penalty |
|---|---|
| Retry after less than 5 s | 30 min |
| Retry after less than 1 s | 2 h |
| Portscan | 3 h |
| Ignoring DNS MX priority information | 3 h |
| Ignoring DNS MX information | 3 h |
| No DNS PTR record, or non-responding DNS server | 6 h |

TABLE II

Penalty Calculation for *mail.gmx.net*.

| Try | Time | $\Delta t$ | n | Penalty | Obs. Per. | Action |
|---|---|---|---|---|---|---|
| #1 | 0 s | - | 0 | 900 s | 900 s | deny |
| #2 | 400 s | 400 s | 0 | - | 900 s | deny |
| #3 | 1200 s | 800 s | 0 | - | 900 s | permit |

For extremely short retry times, i. e., hammering, an additional penalty is incurred. Furthermore, in addition to the retry behavior, our implementation considers some additional sources of information. First of all, it is augmented by classical black- and whitelisting. Hosts that are blacklisted on a DNSBL are never let through, while specific, well-known mail sources are whitelisted; the latter applies in particular to servers that send time-critical email (like, e. g., ebay alerts).

Other criteria for imposing penalties include those that have been used in the previous section to identify bad mail sources in our data set. An overview of the values used in our implementation is given in Table I. Note that the observation period based approach is inherently extensible: if additional information, or just an indication, on the nature of a spam source is available, then it can easily be included by incorporating it in the calculation of the observation period.

We have used the data collected during the MX retry behavior analysis described in Section III to simulate how our filtering algorithm would have reacted to the retry patterns occuring there, and whether its reaction matches our intention. Table II shows the retry pattern of the MX of one of Germany's biggest free-mail providers. This server does not contact the secondary MX at all, but exhibits a well-behaved timing. The table reads as follows: the first line denotes the server's initial connection attempt (#1). Then the observation period is initialized with 900 s. As the second line shows, the server's first reconnect (connection attempt #2) happens after 400 s. Since the observation period has not yet elapsed, but 400 s is greater than $t_{\text{ER}}$, the connection is rejected, but no additional penalty is imposed. The third connection attempt is finally let through, because the observation period has elapsed.

Table III shows the first events during the observation period of a typical spambot. The table shows that the secondary MX is connected first (mx2), and that port scans are detected (ids), causing penalties. The timing is chaotic, early reconnects and also multiple consecutive early reconnects (see column $n$) happen continuously.

We assume that a mail source that made it through the observation period is well-behaved. Thus, such a mail source is

TABLE III

PENALTY CALCULATION FOR *p54830063.dip0.t-ipconnect.de*.

| Try | Time | $\Delta t$ | n | Penalty | Obs. Per. | Action |
|---|---|---|---|---|---|---|
| *mx2* | -5 s | - | - | 3 h | 10800 s | deny |
| #1 | 0 s | - | - | 900 s | 11700 s | deny |
| *ids* | 22 s | - | - | 3 h | 22500 s | - |
| *ids* | 22 s | - | - | 3 h | 33300 s | - |
| #2 | 22 s | 22 s | 1 | 158 s | 33458 s | deny |
| *mx2* | 391 s | - | - | - | 33458 s | deny |
| #3 | 396 s | 374 s | 0 | 0 s | 33458 s | deny |
| *ids* | 417 s | - | - | 3 h | 44258 s | - |
| #4 | 417 s | 21 s | 1 | 159 s | 44417 s | deny |
| *mx2* | 481 s | - | - | - | 44417 s | deny |
| #5 | 486 s | 69 s | 2 | 222 s | 44639 s | deny |
| #6 | 507 s | 21 s | 3 | 477 s | 45116 s | deny |
| *ids* | 508 s | - | - | 3 h | 55916 s | - |
| *ids* | 508 s | - | - | 3 h | 66716 s | - |
| *mx2* | 523 s | - | - | - | 66716 s | deny |
| #7 | 528 s | 21 s | 4 | 636 s | 67352 s | deny |
| #8 | 549 s | 21 s | 5 | 795 s | 68147 s | deny |
| ... | | | | | | |

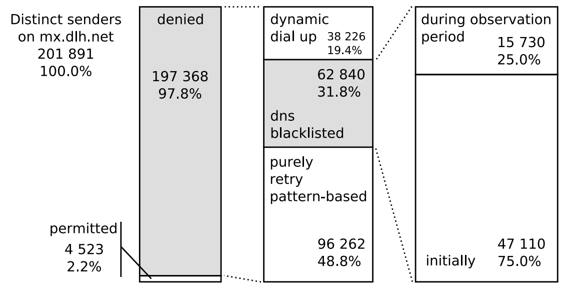

Distinct senders on mx.dlh.net 201 891 100.0%
denied
197 368 97.8%
permitted
4 523 2.2%
dynamic dial up 38 226 19.4%
62 840 31.8%
dns blacklisted
purely retry pattern-based
96 262 48.8%
during observation period 15 730 25.0%
initially 47 110 75.0%

Fig. 2. Statistical breakdown of distinct senders on mx.dlh.net.

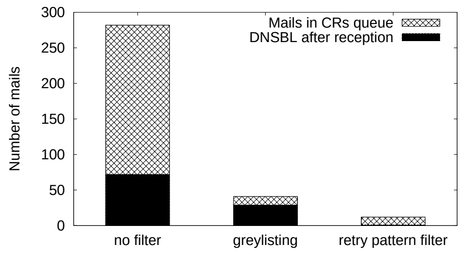


Fig. 3. Mails without CRS response on one mx.dlh.net account (1 day).

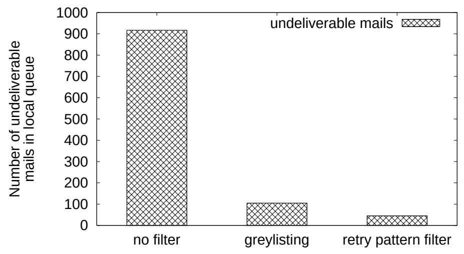


Fig. 4. Undeliverable mails in queue of mx.dlh.net (1 day).

whitelisted for some days or weeks, before it is probed again. Furthermore, mail hosts that have been the destination mail exchanger for mail sent by local users are also immediately whitelisted.

To keep the connection handler proxy as lightweight as possible, time consuming tasks like DNS resolution or DNSBL lookups are performed by an asynchronous worker process. In our implementation, all dynamic information is stored in a database. This allows for easy backups and increases scalability. Moreover, it allows for different connection handler proxies to share information on observed mail source behavior. This is useful in heavy-load setups, where load balancing is used. It also forms the basis of an infrastructure where mail exchangers of different domains might collaborate to identify spambots and well-behaved mail sources more quickly and more precisely, by sharing information on the observed retry behavior.

## V. EVALUATION

We had the opportunity to evaluate our algorithm in production use on three major mail servers since the beginning of July 2006. Here, we discuss the results from the first weeks of its practical use.

Analyzing a pre-MX spam filter is hard, because one cannot tell whether a denied connection would have delivered spam or not. Therefore, we need to rely on indirect evaluations. The subjective impression of the users on the involved mail servers (most of them were not informed about the new filter system) was that the overall spam burden has significantly decreased, although there was already ordinary greylisting installed on that servers before.

Figure 2 provides some hints on the filter performance. It shows a statistical breakdown of the distinct mail sources connecting to one of the evaluation hosts, mx.dlh.net. Only 2.2 % of the senders were permitted at all. Of the rest, 48.8 % never made it through the observation period, and 31.8 % were blacklisted. It is interesting to note that out of the latter, the vast majority of 91.5 % would anyway not have been let in because of their retry pattern.

To gain a deeper insight into the performance of our filter, we have disabled all filters for one week, and compare some statistics to those of the following week with our filter enabled.

On mx.dlh.net there is a challenge-response system installed on most mailboxes. In Figure 3 one can see the number of mails arriving on one day that did not make it through the CRS, for one of the busiest accounts on that system. The number of such mails heavily decreased with our filter, also in comparison to greylisting. The mails still remaining were mainly vacation autoresponder messages or delivery failure notifications sent by regular mail servers. The number of mails whose source was blacklisted after their reception dropped almost to zero. This provides an evidence for us having achieved our goal that our filter delays bad senders long enough to be listed by DNSBLs.

Associated with the number of mails waiting for the CRS response is the number of undeliverable mails caused by CRS challenges, if those are sent to non-existing addresses. This number is also noticeably smaller, as Figure 4 shows.

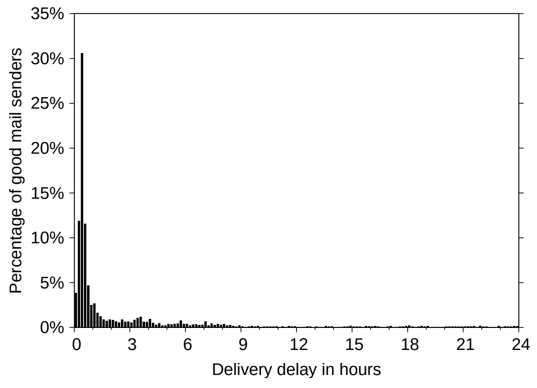


Fig. 5. Delay of first mail delivery.

To check if our filter missed some misbehaving mail sources we also randomly checked some spam mails by hand, if they had been let through. All the respective sources had done proper queueing and timing during the observation period. So, there was no chance for our filter to classify them as bad.

In our approach as well as with common greylisting a delay can occur until a mail is delivered, because the respective mail source is not immediately let through. This delay is, however, much less severe for our approach. To demonstrate this treat of retry pattern-based filtering, we have analyzed how long the delay is in practice. Recall that all the processing is done per mail source. Therefore, different from greylisting that works on a mail address pair basis, only the first mail coming from the same mail server is delayed, while all future ones are delivered immediately—independently from sender and destination email addresses. In our production environment this applies to about 94% of all emails, thus only six percent of the incoming mail is delayed at all. Since our data is from the initial time of use, we expect this number to drop even further, as more good mail sources are being learned over time.

For the six percent of delayed mail, Figure 5 shows a histogram of the delays. It is obvious that the delay is typically very small, even less than one hour for 65 % of the mails. This is below the fixed delay of greylisting in a typical configuration. The long delays typically occur for non-spam mass mailings like, e. g., newsletters. MXs delivering such mail are typically configured for maximum throughput and are thus generally well-behaved, but rather aggressive.

One last, but very important observation is that the average system load on the mail server dropped by almost 50 percent with retry pattern-based filtering. The newly free CPU time can be used for more powerful, computationally expensive content-based filters.

## VI. Conclusion

In this paper we have described a simple, lightweight, but very effective spam filter intercepting the mail delivery process at an early stage. It is based on connection-oriented analysis of mail source retry patterns, observed by temporarily rejecting their delivery attempts. An evaluation revealed that this approach is able to avoid receiving huge amounts of spam at the mail server.

The resources saved by filtering the mail with our approach can be spent for more complex higher level filters. This can increase the probability to identify spam that still managed to find its way through the origin-based filter. As a future extension, it might be possible to utilize the information obtained by observing the retry behavior of a mail source in the content-based filtering. This could yield a holistic spam filtering approach.

Proper queueing implies a relatively high cost at the side of the mail source, if it has to be done for a large number of mails. The key concept of our approach arises from the fact that spammers need to send many spam messages as fast as possible, in order to be effective. So, it is very hard for them to perform RFC compliant timing, without largely decreasing their efficiency. As a consequence, spammers will most likely behave non-RFC compliant for the forseeable future. If they adapt and start using proper timing, their throughput will significantly decrease. If they don't, our filter will be able to identify them with the shown high probability.

## References

[1] Messaging Anti-Abuse Working Group, "Email Metrics Program – 1st Quarter 2006," June 2006. [Online]. Available: http://www.maawg.org/about/FINAL_1Q2006_Metrics_Report.pdf

[2] J. Postel, "Simple Mail Transfer Protocol," RFC 821 (Standard), Aug. 1982, obsoleted by RFC 2821. [Online]. Available: http://www.ietf.org/rfc/rfc821.txt

[3] M. Wong and W. Schlitt, "Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1," RFC 4408 (Experimental), Apr. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4408.txt

[4] J. Lyon and M. Wong, "Sender ID: Authenticating E-Mail," RFC 4406 (Experimental), Apr. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4406.txt

[5] IronPort Systems, "Spammers Continue Innovation: IronPort Study Shows Image-based Spam, Hit & Run, and Increased Volumes Latest Threat to Your Inbox," June 2006. [Online]. Available: http://www.ironport.com/company/ironport_pr_2006-06-28.html

[6] A. Ramachandran and N. Feamster, "Understanding the Network-Level Behavior of Spammers," in *SIGCOMM '06: Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Sept. 2006, pp. 291–302.

[7] E. Harris, "The Next Step in the Spam Control War: Greylisting," Whitepaper, Aug. 2003. [Online]. Available: http://projects.puremagic.com/greylisting/whitepaper.html

[8] P. V. Mockapetris, "Domain names – concepts and facilities," RFC 1034 (Standard), Nov. 1987, updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592. [Online]. Available: http://www.ietf.org/rfc/rfc1034.txt

[9] "Tagged Message Delivery Agent (TMDA)," http://tmda.net/.

[10] B. Agrawal, N. Kumar, and M. Molle, "Controlling Spam Emails at the Routers," in *ICC '05: Proceedings of the IEEE International Conference on Communications*, May 2005, pp. 1588–1592.

[11] D. Cook, J. Hartnett, K. Manderson, and J. Scanlan, "Catching Spam Before it Arrives: Domain Specific Dynamic Blacklists," in *AISW-NetSec '06: Proceedings of the Australasian Information Security Workshop (Network Security)*, Hobart, Australia, Jan. 2006, pp. 193–202.

[12] T. Slettnes, "Spam Filtering for Mail Exchangers," Version 1.0, Linux Documentation Project HOWTO, Sept. 2004. [Online]. Available: http://www.tldp.org/HOWTO/text/Spam-Filtering-for-MX

[13] "DLH.Net." [Online]. Available: http://dlh.net/