Inferring Spammers in the Network Core

Dominik Schatzmann, Martin Burkhart, and Thrasyvoulos Spyropoulos

Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland
{schatzmann,burkhart,spyropoulos}@tik.ee.ethz.ch

Abstract. Despite a large amount of effort devoted in the past years trying to limit unsolicited mail, spam is still a major global concern. Content-analysis techniques and blacklists, the most popular methods used to identify and block spam, are beginning to lose their edge in the battle. We argue here that one not only needs to look into the network-related characteristics of spam traffic, as has been recently suggested, but also to look deeper into the network core, to counter the increasing sophistication of spammers. At the same time, local knowledge available at a given server can often be irreplaceable in identifying specific spammers.

To this end, in this paper we show how the local intelligence of mail servers can be gathered and correlated *passively*, scalably, and with low-processing cost at the ISP-level providing valuable network-wide information. First, we use a large network flow trace from a major national ISP, to demonstrate that the pre-filtering decisions and thus spammer-related knowledge of individual mail servers can be easily and accurately tracked and combined at the flow level. Then, we argue that such aggregated knowledge not only allows ISPs to monitor remotely what their "own" servers are doing, but also to develop new methods for fighting spam.

1 Introduction

According to IronPort's 2008 Security Trend Report [1], as much as 90% of inbound mail is spam today. Moreover, spam is no longer simply an irritant but becomes increasingly dangerous. 83% of spam contains a URL. Thus, phishing sites and trojan infections of office and home systems alike are just one click away. The rapid increase of spam traffic over the last years poses significant processing, storage, and scalability challenges for end-host systems, creating a need to at least perform some fast "pre-filtering" on the email server level. To do this, email servers evaluate information received at various steps of the SMTP session using local (e.g., user database, greylisting [2]) and global knowledge (e.g., blacklists [3,4] or SPF [5]) to identify and reject malicious messages, without the need to look at the content.

Nevertheless, traditional pre-filtering methods like blacklists are starting to lose their edge in the battle. Spammers can easily manipulate an IP block for a short time to do enough damage before they can be reported in a blacklist [6,7]. To amend this, new filtering approaches focusing on general network-level characteristics of spammers are developed [8,9,10,11], which are more difficult for a spammer to manipulate. An example of such characteristics are geodesic distance between sender and recipient [12], round trip time [9] or MTA link graph properties [13,14]. These methods have been shown to successfully unveil additional malicious traffic that slips under the radar of traditional pre-filtering. Yet, they require different amounts of information and processing, ranging from simply peeking into a few entries of the packet header to less lightweight, more intrusive approaches.

Our work is in the same spirit, in that we are also interested in the network-level characteristics of spammers. However, we look at the problem from a somewhat different perspective. Specifically, we look at the problem from an AS or ISP point of view comprising a network with a large number of email servers. We assume that a number of servers in this network (if not all) already perform *some* level of pre-filtering, e.g. dropping a session to an unknown recipient, using a blacklist, or even using sophisticated network characteristics based mechanisms like the one proposed in [12]. This essentially implies that (a) each server is not performing equally "well" in identifying and blocking spammers, and (b) each server has a limited, *local* view or opinion about which senders are suspicious or malicious. In this context, we're interested in answering the following question: *can one use a* 100% *passive, minimally intrusive, and scalable network-level method to (a) infer and monitor the pre-filtering performance and/or policy of individual servers, and (b) collect and combine local server knowledge in order to re-use it to improve server performance?*

Although one could potentially use individual server logs to gain the needed pre-filtering information, in order to collect network-wide spam statistics an ISP would have to gather the logs of all mail servers in the network. As these servers are usually located in many different organizational domains, this is a tedious process that is hindered by privacy concerns of server operators. Instead, *we demonstrate that the pre-filtering decisions of individual servers can be passively and accurately inferred in the network using little flow size information captured in the network core* as illustrated in Fig. 1. Having validated this methodology, we then use it to analyze the incoming SMTP traffic of a major national ISP network with 320 internal email servers. We found that internal servers perform very differently. Some servers accept up to 90% of all incoming SMTP flows, while many accept only $10-20\%$. We look further into the causes of these discrepancies, and after ruling out various "benign" causes, we conclude that many servers in the network seem to be mis-configured or simply under-performing. Based on this, we investigate how and to what extent the *collective* knowledge of well-performing servers could be used to improve the pre-filtering performance of everyone.

Summarizing, our method avoids the cumbersome process of log gathering and correlation. It also requires minimal processing and session information, implying that this method is scalable enough to keep up with the high amount of information constantly gathered at the network core. Finally, it is complementary to recently proposed, sophisticated spam detection mechanisms based on network characteristics, in that the whole system could benefit from such increased capabilities deployed a given server or subset of them.

2 Preliminaries

The email reception process on a server consists of three phases as depicted in Fig. 2, TCP handshake, SMTP email envelope exchange, and email data exchange. Pre-filtering is employed in the second phase: in order to identify and quickly reject malicious traffic

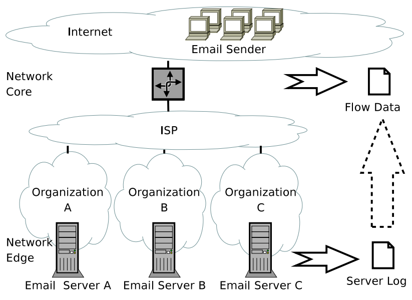

Fig. 1. The ISP view of the network

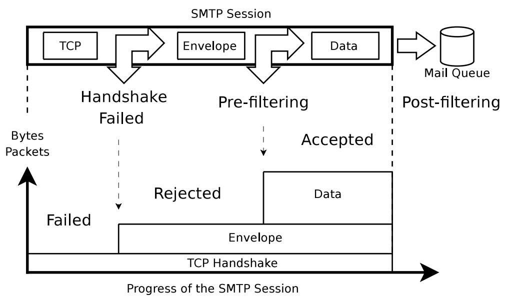

Fig. 2. The three phases of email reception

based on the message envelope only a server may use "global" knowledge (e.g., sender listed in a blacklist), local knowledge (e.g., attempt to reach unknown recipients), or policy-based decisions (e.g., greylisting).

We analyzed the log of a university mail server serving around 2400 user accounts and receiving on average 2800 SMTP flows per hour to look into such pre-filtering performance in more detail. We found that as much as 78.3% of the sessions were rejected in the pre-filtering phase. 45% of the rejects were based on local information (e.g., user database or greylisting) and only 37.5% were due to blacklists. This illustrates the importance of local mail server knowledge for spam detection.

Based on the server's decisions, we classify observed SMTP sessions as either *failed*, *rejected* or *accepted*. Our key observation is that, whenever a sender manages to get to the next phase, the overall transferred information is significantly increased. For example, if a sender is accepted and allowed to send email content, he is able to transmit much more data than a sender already rejected in phase two. As a consequence, we conjecture that flow properties reflecting the size or length of SMTP sessions, such as the flow size or packet count, should be an accurate indicator for the phase in which an SMTP session was closed.

We validate this assumption in Section 3. For this purpose, we have used three weeks of unsampled NetFlow data from January, February and September 2008 (referred to as week 1, 2, 3), captured at the border routers of a major national ISP [15] serving more than 30 universities and government institutions. The IP address range contains about 2.2 million internal IP addresses and the traffic volume varies between 60 and 140 million NetFlow records per hour. The identification of SMTP traffic is based on TCP destination port 25[1]. Based on the SMTP traffic, a list of active internal email servers was generated and verified by active probing. We detected 320 internal servers, receiving up to 2 million SMTP flows per hour.

3 SMTP Flow Characteristics

In this Section, we demonstrate how the effect of pre-filtering on flow characteristics can be used to track the servers' decisions for each SMTP session.

[1] Note that only the traffic flowing from external SMTP clients to internal servers is considered.

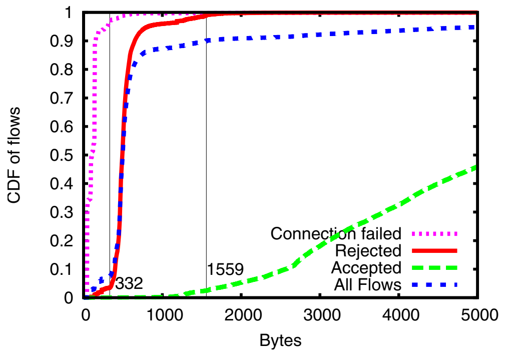

Fig. 3. Byte count distribution

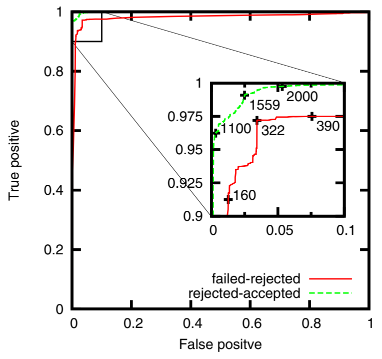

Fig. 4. ROC curve for bytes per flow metric

Table 1. Classification performance for x bytes per flow

	$x < 322$	$322 <= x <= 1559$	$x > 1559$
Failed	9302 (95.64%)	417 (4.28%)	7 (0.07%)
Rejected	11008 (3.59%)	409675 (96.66%)	3132 (0.74%)
Accepted	55 (0.09%)	1662 (2.74%)	58845 (97.16%)

The CDF of byte counts for flows arriving at the mail server in week 1 is presented in Fig 3. The class of failed connections mainly consists of very small flows as only a small number of packets could be sent. 97% of these flows have less than 322 bytes. The size of most rejected flows is between 400 and 800 bytes. This corresponds to the size of the SMTP envelope. Lastly, the distribution of accepted flow sizes is dominated by the overall email size distribution and reaches 50% at around 5000 bytes. This is consistent with the findings of Gomes et al. [16]. The CDF for "all flows" in Fig. 3 is a superposition of the three classes weighted by their relative probability of appearance. All three classes are well visible in the total CDF even though it is dominated by rejected flows due to the fact that around 80% of all flows are rejected.

Next, we determined two optimal threshold sizes to differentiate between *rejected*, *failed* and *accepted* flows. For this purpose, we constructed ROC curves [17] which plot the true positive versus the false positive rate of a detector for a range of thresholds. Fig. 4 shows the ROC curves for the detection of rejected vs. failed and accepted vs. rejected flows. The three classes are distinguishable with high precision. We selected the two thresholds 332 Bytes (rejected vs. failed) and 1559 Bytes because these points are closest to the top left corner and hence yield the best detection quality [17].

We evaluated the false positive rate of these threshold detectors on data of another week (week 2) and present the results in Table 1. The false detection rate is below 4.5% for all classes which is sufficiently accurate for the applications outlined in Section 4[2]. We also analyzed the power of other flow properties to discriminate between the three classes. In addition to *bytes per flow*, also *packets per flow* and *average bytes per packet* are well suited for this purpose [18].

[2] The flow labels assigned by our system are to be treated mostly as "soft" labels. Further accuracy could be achieved by using e.g. clustering algorithms on additional flow fields.

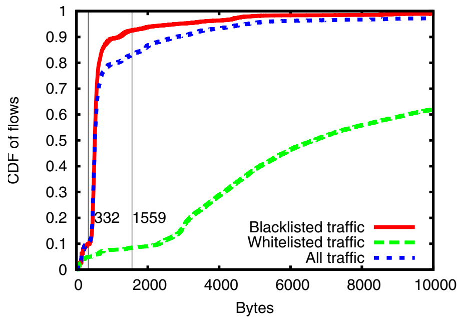

Fig. 5. Network-wide flow sizes

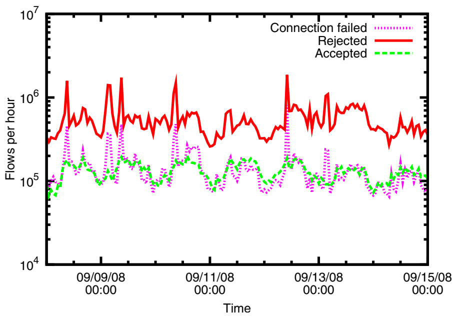

Fig. 6. Network-wide pre-filtering statistics

It is important to note that packet sampling would affect our approach. Over 90% of the rejected SMTP sessions consist of 10 or less packets and more than 90% of accepted sessions have less than 40 packets. With a sampling rate of 1:100 or even 1:1000, the resulting flows would mostly consist of 1 or 2 packets. This would weaken the usefulness of the bytes and packets per flow metrics; yet, our analysis suggests that it could still be possible to distinguish between rejected and accepted flows using *average bytes per packet* [18]. Further, adaptive sampling techniques are being developed [19] that could perhaps address this problem also. We intend to look further into the issue of sampling in future work.

Network-wide characteristics. The classification of flows based on their size allows to passively monitor pre-filtering activity in large networks in a scalable manner, without resorting to server logs. To validate that the observed characteristics also hold on a network-wide scale, we show the characteristics of black- and whitelisted traffic for the 50 most active mail servers in our network in Fig. 5. The shape of black-/whitelisted curves nicely reflects the characteristics of rejected and accepted flows from Fig. 3. Hence, (i) the vast majority of traffic from blacklisted hosts is rejected by our network in pre-filtering and (ii) we are able to infer this reject decisions from flows sizes only. Individual server performance differences are addressed in detail in Section 4.1.

The generation of network-wide pre-filtering statistics, as illustrated in Fig. 6, allows to easily estimate the amount and track the dynamics of incoming spam at the ISP level. An ISP is now able to investigate the root cause of anomalies in rejected/accepted traffic. Potential causes are global spam campaigns that are visible on many servers, spamming attacks targeted to a small set of servers or misconfiguration and performance problems of single servers.

4 Applications

We now turn our attention to potential applications of our method. In Section 4.1, we demonstrate how it can be used to passively analyze the configuration of mail servers and troubleshoot misconfigured servers. We then explore the feasibility and potential of a collaborative filtering system among the various mail servers in Section 4.2.

4.1 Email Server Behavior

Today, adequate configuration and maintenance of mail servers is a time-consuming process. It would be very helpful for operators to get a performance map of the various mail servers present in the network. The state of pre-filtering deployment in the network could be checked regularly and potential configuration problems, (e.g., the presence of open relay servers), could be addressed proactively.

To compare the pre-filtering performance of internal servers, we define the *acceptance ratio* of a server to be the number of accepted SMTP flows divided by the number of total SMTP flows seen by the server. A high ratio of, for example, 0.9 indicates that 90% of all incoming SMTP sessions are accepted, whereas a low ratio indicates that most of the connections are rejected during the TCP handshake or the SMTP envelope. Clearly, the observed acceptance ratio for a server is affected by two parameters: (i) the *traffic mix* of ham and spam for this server, and (ii) the server *prefiltering policy*. To address the former, we estimated the spam/ham mix ratio for each server with the help of the XBL blacklist from Spamhaus. Our analysis shows that spam (flows from blacklisted sources) is evenly distributed among servers. 81% of the servers have a spam load between 70% and 90%, consistent with [1]. This results implies that *big differences in servers' acceptance ratios cannot be attributed to different traffic mixes.*

The server policy issue is somewhat trickier. The above numbers imply that, if all servers were at least using a blacklist, the acceptance ratio of most internal servers should be between 0.1 and 0.3, with differences attributed to traffic mix and sophistication and/or aggressiveness of pre-filtering policies (e.g., greylisting, etc.). Instead, the acceptance ratios of the top 200 servers for week 3 of our data set range from 0.003 up to 0.93 with a mean of 0.33 as can bee seen in Fig. 7. 35% of the servers have an acceptance ratio > 0.30. Based on the above traffic mix estimation, we conclude that they are accepting a lot of traffic from spam sources. This could imply: (i) a regular server that is sub-optimally configured, lacks sophisticated or even simple pre-filtering measures (e.g., lack of time, caring, or knowhow), and/or should at least raise an eyebrow; or (ii) a server whose intended policy is to accept all messages (e.g., servers that apply content-based filtering only, honeypots, etc.)

To verify this assumption, we sent emails to all servers from two different IP addresses: an address blacklisted by Spamhaus and Spamcop and an ordinary, not blacklisted address[3]. The reaction of the servers to our sending attempts clarified whether the server was using greylisting and/or blacklisting. The servers classified as 'unknown' are those servers for which the reaction was not conclusive. The high concentration of black- and greylisting servers below the average ratio shows that, indeed, these servers implement basic pre-filtering techniques, whereas servers that do not implement them lie mostly above average. Also, with increasing volume (to the right), servers with high acceptance ratios tend to disappear. This affirms that administrators of high-volume servers (have to) rely on aggressive pre-filtering to master the flood of incoming mails. We also manually checked high acceptance servers and found no honeypots trying to deliberately attract and collect spam.

[3] It is important to stress that this, and other "manual" investigations we performed in this section are only done for validation purposes, and are not part of the proposed system.

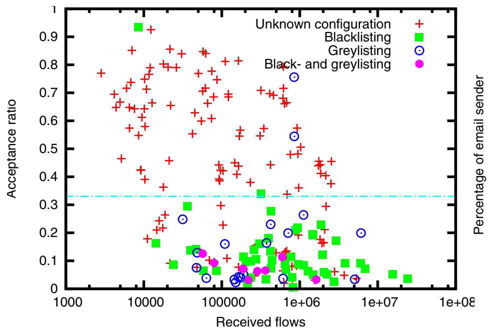

Fig. 7. Server acceptance ratios vs. traffic volume

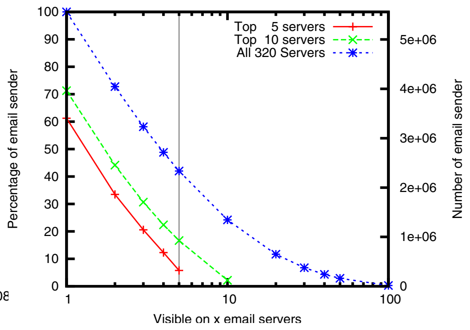

Fig. 8. Visibility of the email senders

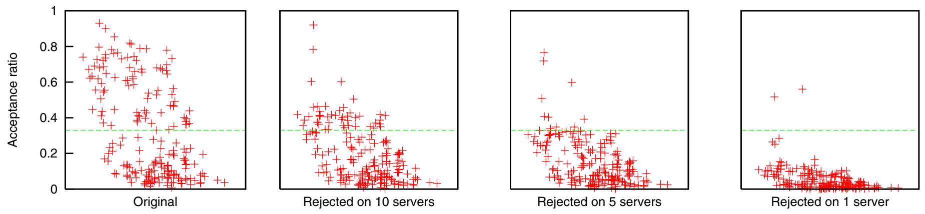

Fig. 9. Improvement potential when using collaborative filtering

We conclude that differences in acceptance ratio are mainly due to configuration issues and that there is a large group of servers that might need a wake-up call or could profit from the expertise of other servers.

4.2 Collaborative Filtering

Given the above observations of unbalanced mail server performance, what could an ISP do to improve overall pre-filtering performance in its network? Surely, the ISP could contact individual administrators and organize a meeting to present the statistics about mail server performance where administrators would exchange their knowhow. However, this requires a lot of organizational effort, needs to take place regularly, and attendance of administrators is not guaranteed. Furthermore, educating customers' email administrators is usually not the ISP's business.

Therefore we investigate a passive way enabling *all* servers to profit from the local knowledge and the best existing anti-spam techniques already present in *some* servers in the network. By accepting more or less emails from a client, email servers actually perform an implicit rating of the very client. With our method, these ratings could be extracted from traffic and used to build a collaborative filtering system (CFS), or more properly, a collaborative rating/reputation system. The system would recommend accepting/rejecting mails from a client, based on the behavior of the collective of all servers: "This host was rejected by 80% of the servers in the network. Probably you should reject it as well."

It is important to note that the added value of the collected information depends on the ability of different servers to block different flows: heterogeneous knowledge or heterogeneous pre-filtering detection techniques and policies are *desirable* since this increases the chance that at least some methods will detect the spammer.

At the same time, a significant overlap of the sets of email senders visible to each server is needed, in order to achieve useful and trusted enough ratings. We analyzed the visibility of the sending hosts on three sets of top 5, top 10 and all internal servers (see Fig. 8). More than 5.5 million email senders are visible on the 320 internal servers. Moreover, 72% of all email senders are visible on at least two internal servers. This means that for 72% of the email senders a second option from another server is available. Note that even the top 5 email servers only see 61% of the sending hosts. In addition, only 6% of the email senders are visible on all of these five servers. This percentage is increased to 17% for the top 10 servers. By using information from all internal servers, 42% of the sending hosts are visible on at least 5 different servers, which is a promising foundation for building a CFS. In addition we explicitly analyzed the visibility of hosts that are listed in a black- or whitelist. The visibility of blacklisted hosts follows the overall host visibility. However, the visibility of whitelisted hosts is even better (e.g., 60% of the whitelisted hosts are visible on at least 5 different servers).

The actual implementation of a CFS is beyond the scope of this paper. Nevertheless, we are interested here in estimating the potential of such a system by simulating simple filter rules. Specifically, we simulated an *online* version of the system where blocklists are created as incoming flows are parsed according to some simple rules. Specifically, counters are maintained for senders that have been consistently blocked with the following 3 rules:[4] As soon as a sender's connections have been blocked by at least 10, 5 or 1 server(s), respectively, the sender is entered into our blocklist. Then, if a sender is on the list all incoming connections by this sender will be counted as rejected. The resulting acceptance ratios assuming all servers are using the CFS are shown in Fig 9.

There is an inherent tradeoff in the number of server "votes" used to make a decision. By requiring many rejecting servers (e.g., 10) for membership in the blocklist, the reliability of the filtering becomes quite strong (only 3 blocked hosts were actually whitelisted). Yet, the size of the blocklist is reduced, and provides less additional information than blacklists. Specifically, in the 10 server case, the blocklist has 83% overlap with the Spamhaus blacklist and could be used to block 47% of all SMTP sessions. In the other extreme, if only one rejecting server is required to be put on the blocklist, acceptance ratios of all servers are dramatically reduced. Further, the blocklist overlap with Spamhaus is only 70% and could be used to block up to 95% of all SMTP connections. However, 185 members were whitelisted. That is, requiring only one rejecting server introduces a higher false rejection rate. Nevertheless, it is important to note that in both cases, a significant amount of hosts identified by this method are *not* found in the blacklist, underlining the collaborative nature of the system, and implying that simply making under-performing servers use a blacklist, would not suffice.

Concluding, our estimation shows that there is a significant information overlap that can be leveraged to improve overall pre-filtering by making local server information accessible to the entire network, in an almost seamless manner. Today, pre-filtering is dominated by the use of DNSL blacklists. However, the CFS is independent of the very techniques applied and will automatically profit from upcoming improved techniques.

[4] To reduce the effects of greylisting we delayed the blacklisting process by 15 min.

As a final note, due to the inherent "softness" of the labels attached by such a flow-based system, and the various risks of blocking a session in the network core, we stress here that our system is not intended as an actual blocking filter, but rather as a reputation rating system, which individual email servers can *opt* to include in their pre-filtering phase. Consequently, servers whose policy is to accept all emails, do not have to be affected by our system, although using it, could perhaps provide hints to the content-based filter.

5 Discussion

Although being only a first step, we believe the proposed method has important potential to be applied in production networks and also paves the way for future research in the area of network-level and network-wide characteristics of spam traffic. In this Section we discuss some possible limitations of our approach.

Delay: The flow data is exported by the router only after the TCP connection is completed. Therefore, the information about the flow is delayed at least until the session is closed. We measured this delay to be less than 7.7 seconds for 90% of all SMTP flows. This illustrates that any application based on flow characteristics is limited to near-realtime. In particular, properties of a flow can not be used to intercept this very flow. For our proposed applications, this limitation is acceptable as we are not interested in using the properties of a flow to intercept this very flow, but rather subsequent ones.

Flow size manipulation: In principle, spammers could adapt to our method by prolonging the envelope phase, for instance by sending multiple RCPT or HELO commands. This would indeed increase the number of transmitted bytes per flow but will, at the same time, increase the number of packets. However, the average number of bytes per packet remains small and the bytes per packet metric could still be used for the classification. Further, the spammer could try to increase the bytes transmitted in each command by using long email addresses, but maximum size for a command is limited to 521 characters [20]. Moreover, any deviation from normal SMTP behavior could easily be detected and mail servers could enforce the shortness of the envelope phase as there is no need to be overly long in a normal use case. In addition, the misbehavior of the host could be published to make this information available for other servers.

In a more sophisticated scenario, a spammer could take over different internal hosts and reconfigure them as internal mail servers. By sending accepted SMTP traffic from bot to bot, one could try to positively influence the CFS ratings for these bots. The design of the CFS needs to be aware of this problem. In a first step, only servers that have been active over a longer time period (i.e., they have been accepting mails for at least several weeks) and get a certain amount of connections from trusted email servers (e.g., on a whitelist) could be included into the filtering process.

6 Conclusion

Mail server administrators are engaged in an arms race against spammers. They urgently need new approaches to fight state-of-the-art attachment spam increasingly originating from low-profile botnet spammers. In this paper, we demonstrated that simple flow metrics, such as byte count, packet count, and bytes per packet, successfully discriminate between spam and ham flows when pre-filtering is deployed in mail servers. Thus, one could infer individual mail server's decisions with respect to the legitimacy and acceptance of a given connection. This allows an operator i) to concentrate dispersed mail server knowledge at the network core and ii) to passively accumulate network-wide spam statistics, profile filtering performance of servers, and rate clients. Thus, the advantages of flow and server log analysis finally meet at the network core. We believe this is an important step towards successfully fighting spammers at the network-level.

References

1. IRONPORT: 2008 internet security trends, `http://www.ironport.com`
2. Harris, E.: The next step in the spam control war: Greylisting (2003)
3. SpamCop: Spamcop blocking list, `http://www.spamcop.net/bl.shtml`
4. Spamhaus: The spamhaus block list, `http://www.spamhaus.org/sbl`
5. Wong, M., Schlitt, W.: Sender Policy Framework (SPF). RFC 4408
6. Ramachandran, A., Dagon, D., Feamster, N.: Can DNS-based blacklists keep up with bots. In: Conference on Email and Anti-Spam, CEAS 2006 (2006)
7. Duan, Z., Gopalan, K., Yuan, X.: Behavioral Characteristics of Spammers and Their Network Reachability Properties. In: IEEE International Conference on Communications, ICC 2007 (2007)
8. Ramachandran, A., Feamster, N., Vempala, S.: Filtering Spam with Behavioral Blacklisting. In: ACM conference on Computer and Communications Security, CCS 2007 (2007)
9. Beverly, R., Sollins, K.: Exploiting Transport-Level Characteristics of Spam. In: CEAS 2008 (2008)
10. Clayton, R.: Using Early Results from the spamHINTS. In: CEAS 2006 (2006)
11. Gu, G., Perdisci, R., Zhang, J., Lee, W.: BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection. In: USENIX Security Symposium (July 2008)
12. Syed, N.A., Feamster, N., Gray, A., Krasser, S.: Snare: Spatio-temporal network-level automatic reputation engine. Technical Report GT-CSE-08-02, Georgia Tech. (2008)
13. Desikan, P., Srivastava, J.: Analyzing network traffic to detect e-mail spamming machines. In: ICDM Workshop on Privacy and Security Aspects of Data Mining (2004)
14. Gomes, L.H., Almeida, R.B., Bettencourt, L.M.A., Almeida, V., Almeida, J.M.: Comparative Graph Theoretical Characterization of Networks of Spam and Legitimate Email. Arxiv physics/0504025 (2005)
15. SWITCH: The swiss education and research network, `http://www.switch.ch`
16. Gomes, L.H., Cazita, C., Almeida, J.M., Almeida, V., Meira, W.: Characterizing a spam traffic. In: ACM SIGCOMM conference on Internet measurement, IMC 2004 (2004)
17. Fawcett, T.: An introduction to roc analysis. Pattern Recognition Letters 27 (2006)
18. Schatzmann, D., Burkhart, M., Spyropoulos, T.: Flow-level characteristics of spam and ham. Technical Report 291, Computer Engineering and Networks Laboratory, ETH Zurich (2008)
19. Ramachandran, A., Seetharaman, S., Feamster, N., Vazirani, V.: Fast monitoring of traffic subpopulations. In: ACM SIGCOMM Conference on Internet Measurement, IMC 2008 (2008)
20. Klensin, J.: Simple mail transfer protocol. RFC 2821 (April 2001)