Performance Analysis of a Lightweight Spam Restraining Method on the Sender Side

Jochum Börger
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
j.m.borger@student.utwente.nl

ABSTRACT

A large part of email traffic on the Internet consists of spam. The mail servers of Internet Service Providers could receive a bad reputation as spam is send from them. A bad reputation may lead to legitimate email being blocked. Stopping spam at the receiver end will not solve this problem and only has effect after the spam email has already generated Internet traffic. Both issues can be reduced by filtering outbound spam email.
In 2010 de Vries et al. [20, 21] proposed a solution to stop spam at the sender side using lightweight techniques. The solution showed great potential in an experimental setup, but was not implemented and tested in a real life environment.
This paper contributes to reducing the email spam problem by showing that the promising techniques proposed by de Vries et al. do not get good results in a real life environment and suggesting an alternative solution, showing potential from the results obtained by using the techniques proposed by de Vries et al. as a primary filter and SpamAssassin as a secondary.

Keywords

Lightweight, Spam, Filtering, Comparison

1. INTRODUCTION

In 2012 72,1% of email traffic on the Internet consisted of unsolicited bulk email, or spam [9]. An Internet Service Provider (ISP) can be a source of spam as end-users of their mail servers could be compromised. The amount of spam sent by an email server will determine its reputation. A bad reputation may lead to the server being blacklisted and, as a consequence, legitimate email being blocked. Being a source of spam emails is expensive for ISPs [15].

ISP have been adopted measures to minimise spam into mail servers. Blacklists and spam filters are examples of these measures. Another measure also adopted is called Domain Key Identified Mail (DKIM). This measure employ signature policies to help verify the domain from which emails are sent [9]. These anti-spam measures effectively help filter out spam email, but at the receiver side. It means that spam emails continue traveling and consuming network resources. One way to secure the reputation of email servers and minimise the consuming of network resources is to filter outbound spam email.

There are few solutions available to filter spam in outbound emails. Commercial solutions, for example MailChannels [12] and Commtouch [5], are easy to setup, but expensive to acquire and maintain. On the other hand, SpamAssassin, an open source solution, has a reputation of being a relatively heavyweight [13], and as a consequence it needs to run on powerful hardware. Therefore, considering both commercial and heavyweight solutions, most ISPs do currently not filtering outbound emails [7], because available solutions can be expensive to purchase or in hardware usage, respectively. In order to fulfil the need for an inexpensive lightweight solution de Vries et al. proposed the usage of a number of easy and cheap techniques. These techniques were tested separately on Email Feedback Reports (EFR), and showed great potential [20, 21]. The proposed techniques however were not implemented and tested in an active email environment of an ISP. As a result the real-time performance and effectiveness could not be determined and the lightweight classification not be confirmed. Therefore the usability in a real life environment is not proven.

In this paper the anti-spam techniques proposed by de Vries et al. are further researched. The main research question associated with this paper is:

- *Can lightweight techniques used to restrain the transmission of spam email in a real life environment operate in a more effective and efficient way than existing techniques?*

In order to answer the above research question the techniques are analysed based on effectiveness and efficiency. Effectiveness is a performance metric that show how well the techniques are able to detect and mark spam emails. This metric is measured by using percentages that are calculated by dividing the number of the detected spam emails by the total number of observed emails. Efficiency is a relative metric calculated by measuring the different aspects of hardware utilisation of the techniques.

To answer the main research question the following subquestions have been defined:

1. *Which techniques can be used to restrain transmission of spam email in a real life environment?*
2. *Which variables determine whether a technique is lightweight by comparison?*
3. *What is the effectiveness and efficiency of a lightweight solution in comparison to an existing (non-)lightweight solution in a real life environment?*

19th Twente Student Conference on IT June 24th, 2013, Enschede, The Netherlands.

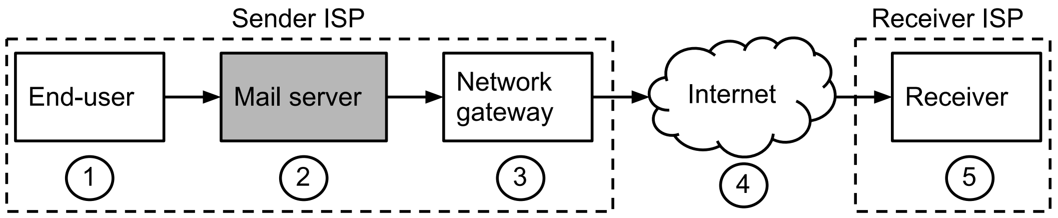

Figure 1. Spam stopping points, based on [2]

In order to evaluate the effectiveness and efficiency of the techniques a prototype has been developed within an active email environment of an ISP. Furthermore, an alternative and existing outbound spam filter was setup within the same environment. The results of both the prototype and the alternative setup were compared and analysed using the effectiveness and efficiency metrics. A positive answer to the main research question would suggest that the proposed techniques can be used by every ISP. The achieved results were unexpected and show that the lightweight techniques are performing well in a lab environment, but not in a real life environment.

The rest of this paper is organised as follows: in Section 2 the techniques for restraining outbound spam are explained, answering *question 1.* Section 3 describes the proposal and which variables of these techniques can be used to compare the efficiency, *question 2.* In Section 4 the setup of the case study is illustrated. Section 5 contains the analyses of the results, together with Section 4 answering *question 3.* Finally Section 6 contains the conclusions of the research and provides recommendations for future work.

2. TECHNIQUES FOR RESTRAINING SPAM TRANSMISSION

Ample research has been done in spam filtering techniques [2, 3, 4, 6, 17]. Spam email can be restrained on five main points in an email transaction as seen in Figure 1. First, see spam stopping point (1) in Figure 1, before the email reaches the sending mail server, at the client side. Second, see spam stopping point (2) in Figure 1, prior to the spam email being sent, using filters on the mail server or within a smart host just after the mail server. Also the mail server logs can be analysed [3, 4]. Third, see spam stopping point (3) in Figure 1, leading up to the spam email leaving the network, for example by applying audit log analysis [6] or analysing network flow data [17]. Fourth, see spam stopping point (4) in Figure 1, on the Internet, by ISPs between the sender and receiver. Or, finally, see spam stopping point (5) in Figure 1, at the receiver end.

Commonly spam email is stopped at spam stopping point (5) and as a consequence has already generated Internet traffic. Therefore, in this paper techniques usable at the Sender ISP are researched. Specifically at stopping point (2) as this point is already used for processing emails by the ISP. Techniques using log analyses are not considered since these techniques mainly identify compromised hosts and therefore, they are less usable for real time spam filtering. In the context of this paper four techniques proposed by de Vries et al. are explored that can be used to stop spam before it reaches the Internet at spam stopping point (2). These techniques make use of the Sender Policy Framework (SPF), URL blacklists, Google Safe Browsing blacklist and FROM-address verification.

De Vries et al. performed effectiveness measurements on these four techniques making use of emails marked as spam. The four techniques were conducted on these emails and the percentage of emails hit by these checks were computed as shown in Table 1 [20, 21].

Table 1. Measurements by de Vries et al.

Technique	Percentage of spam hit
Sender Policy Framework	38.6%
URL blacklist	74.4%
Google Safe Browsing	50.9%
FROM-address verify	100%

Except for the URL blacklist technique all checks were performed on AOL Email Feedback Reports, emails manually classified as spam by AOL users. The URL blacklisting was determined by analysing the SpamAssassin logs on selected email servers. As the URL blacklist checks were performed on a different range of spam emails there is no percentage available for the success rate of the techniques combined.

These results are very promising and further research to discover if these techniques can be valuable in a real life environment is relevant. Therefore in the next section the techniques used by de Vries et al. are further described. Also SpamAssassin is described in this section as this software is used to as a comparative technique to the techniques used by de Vries et al..

2.1 Sender Policy Framework

The Sender Policy Framework (SPF) can be used to detect email spoofing by verifying the sender IP address. Spoofing is an alteration to the email-headers. The FROM-header can be altered by the sender of the email in such a way that it appears as if the email originates from a different location. An SPF check contains of two simple and fast steps; determine the IP address from which the email was sent and look up the SPF record in the DNS for the host used. A comparison between the two shows the validity of the hostname [22].

Every working domain name has DNS records available for the whole Internet. However, the SPF record is not essential and not available for every domain name. As the hostname to be verified should have DNS records within the ISPs network we can make sure an SPF record is available for every valid domain name in within the ISPs network. This check can be used at every spam stopping point in an email transaction, see Figure 1.

2.2 URL blacklist

The URL blacklist is a list of URLs known to be used for spam purposes. These are constantly updated with new URLs that were reported as being used in spam emails. As the target of spam email is to make money the message must contain a referral to a location on which this can be obtained. As a result it can be deemed that probably the message contains an URL.

The URL blacklist works as an DNS server. A hostname can be looked up using this DNS server and the response shows if the hostname is listed. Several URL blacklists are available containing URLs considered malicious or known to be used by spammers. Examples of URL blacklists are URIBL [19] and SURBL [18]. As these blacklist use the DNS system for lookups they are simple to implement and fast in usage. This check can be used at spam stopping points (1), (2) and (5), see Figure 1.

2.3 Google Safe Browsing

The Google Safe Browsing (GSB) initiative is an alternative URL blacklist containing a real time database URLs suspected of containing malware of phishing. New URLs are added by Google when, while inspecting the source of a crawled page, code snippets are found known to be malicious. Google crawls some pages more frequent than others, so it can take some time before an infected page is added to the blacklist. Another difference with the standard URL blacklist is that full URLs to pages are used instead of hostnames.

Lookups to this database can be done using the Safe Browsing API over HTTP that Google had made freely available. The API also provides the possibility to keep a local database and update periodically, to shorten lookup times. Implementation may however be complex [8]. The check is usable at the same spam stopping points as the URL blacklist check.

2.4 FROM-address verify

The FROM-address check verifies if the FROM-address specified in the headers exists. The host of the FROM-address is verified using the SPF check. Regardless of the result, the FROM-address might still be non-existent, as the user-part of the address is not verified. To determine if the sender address exists, and thereby partly the spam probability, the mail server for the email address host can be contacted.

The Simple Mail Transfer Protocol (SMTP) provides a command to verify an email address (VRFY), but unfortunately this command is not supported by a significant percentage of mail servers. Alternatively the SMTP command used to specify the recipient address (RCPT TO) can be used, and the validity of the address can be determined by the server response [11].

Using SMTP to verify an address can be prone to induce delays, as the server is slow or might not exist. This check might also be considered abusive [1], and de Vries et al. recommends to only perform this check if the other checks do not indicate spam [20, 21].

2.5 SpamAssassin

SpamAssassin is an open-source anti-spam software. It uses a number of different techniques to calculate the spam probability to an email. Examples of techniques used by SpamAssassin are header analyses, as a SPF check, content analyses, Bayesian filtering, DNS- and URL- blacklists. All techniques used by SpamAssassin add a positive or negative number to the spam score of an email. The higher the score the more likely SpamAssassin assess the email to be spam [16].

In the default configuration of SpamAssassin emails scored 5.0 or higher are considered spam. This number is judged to be quite aggressive [16]. When referring to email marked as spam by the SpamAssassin setup, the email is then scored 5.0 or higher.

SpamAssassin can be used as an extension to many different SMTP daemons [16], it can, for example, be used as a plug-in to the SMTP daemon *qpsmtpd* [14].

3. PROPOSAL

The techniques described in Section 2 and proposed by de Vries et al. in 2010 [20, 21] were not yet tested in a real life environment although the results acquired from experimental testing showed great potential. Furthermore is the experimental setup the techniques were not combined. In this research a case study within a real life environment is performed where the four techniques are combined and compared to an alternative technique on effectiveness and efficiency.

A prototype has been implemented using the four described techniques. In the context of this paper we denote our prototype as Prototype. For comparison SpamAssassin has been chosen, since it is an open source solution, and thus freely available to use in these experiments, easy to setup, and gives great insight in the spam scores set so further analyses can be performed [16]. The Prototype as well as the SpamAssassin setup are used within a real life environment.

The two spam stopping solutions are compared on several aspects. To determine the difference in effectiveness of both the Prototype as SpamAssassin the spam probability of emails processed by these setups is checked, the hits on the four Prototype techniques and the spam score of SpamAssassin are combined for every email scanned. This metric is measured by using percentages that are calculated by dividing the number of the detected spam emails by the total number of observed emails. The efficiency is compared by monitoring the different aspects of hardware utilisation of both setups, as well as determining the process time of every email scanned on both setups.

The privacy of the end-users of the cooperating ISP must be safeguarded, the emails may not be inspected manually. As a consequence in these experiments only the results of the two setups can be compared and contradictions cannot be further analysed. The results are collected in a database.

De Vries et al. defined several requirements. Two of these requirements were targeted on the performance of the implementation regarding the hardware, and the maximum effort it should take to produce a working implementation [20, 21]. As no implementation was produced they were unable to confirm that these requirements were met.

One requirement set by de Vries et al. was that it should be possible to implement the techniques within 200 hours [20, 21]. This requirement was met easily with more than half of the available hours to spare.

For the second requirements de Vries et al. stated that the techniques should not be allowed to use more than 5% of the CPU capacity [20, 21]. The requirement is however interpreted as meaning that the techniques should be lightweight. Lightweight is a relative term classifying a process as being not resource intensive. This term is relative as the available capacity can differ. A process using a small percentage of the hardware resources within one system can be considered a lightweight process on this system, while it may use a large percentage of the resources on another system containing less available resources.

Jain (1991) describes a number issues associated with capacity planning [10]. There is for example no standard definition for capacity or for a workload unit. Moreover, one system can contain a number of different capacities, e.g. for memory and storage. In this paper the term lightweight will be used to relate the performance of one solution compared to another one. It will be determined if the proposed techniques are lightweight relative to the compared techniques and the setups used.

Furthermore, it is considered that the configurations of both Prototype and SpamAssassin are done in such a way that they can have the same amount of resources available.

4. CASE STUDY

To perform the case study two proxy mail servers are configured within the ISPs email platform to which the duplicate emails are routed. One proxy is running the Prototype and the other proxy is running SpamAssassin. Together they are creating the configuration shown in Figure 2 in an ISPs email platform. In this section the tools and specifics of the case study are described.

The measurements of this case study are performed during 72 hours on 24th, 25th and 26th of april 2013, scanning a total of 69535 emails.

4.1 Environment

The case study is performed within an ISPs network, the Dutch hosting provider Antagonist B.V. Email is sent by 14 mail servers, running Exim, an SMTP daemon. The two proxy mail servers are configured to accept every email routed from all servers within the same network. These are configured to route duplicates of all emails to both proxy servers using manual routes in the mail server configuration of these servers. For both proxy servers the new route was added to the Exim configuration as follows:

```
custom_setup_route:
  driver = manualroute
  transport = remote_smtp
  route_list = !+local_domains proxyserver.tld
  unseen
```

The unseen option[1] set at the end ensures the mail server does not stop processing after it encounters this appropriate route for delivery, resulting in duplicate emails. The Prototype and SpamAssassin system process exactly the same emails. As a result an email sent by an end-user or spammer is delivered as normal, but also scanned by two proxy servers.

4.2 Tools

The two proxy servers are two virtual machines made available for the case study. CentOS 6.4 is the used as the operating system. On both systems *qpsmtpd 0.84* was installed, this is an SMTP daemon written in Perl as a drop-in qmail-smtpd replacement [14]. This SMTP daemon is used because it has an advanced and simple to use plugin system, furthermore numerous[2] useful plugins are already available, open source and written in Perl.

For the Prototype two plugins included in qpsmtpd are used. The *uribl* plugin configured to use the URIBL and SURBL blacklists, and write its results in an email header. Also the *sender_permitted_from* plugin is used for the SPF check, slightly edited to accept relaying and add an email header with the result. The GSB technique is implemented using the Perl library *Net::Google::SafeBrowsing2*. The check uses a locally stored blacklist for speed, updated every thirty minutes via the HTTP API. Finally,

[1]`http://www.exim.org/exim-html-current/doc/html/spec_html/ch-generic_options_for_routers.html`
[2]`http://wiki.qpsmtpd.org/plugins`

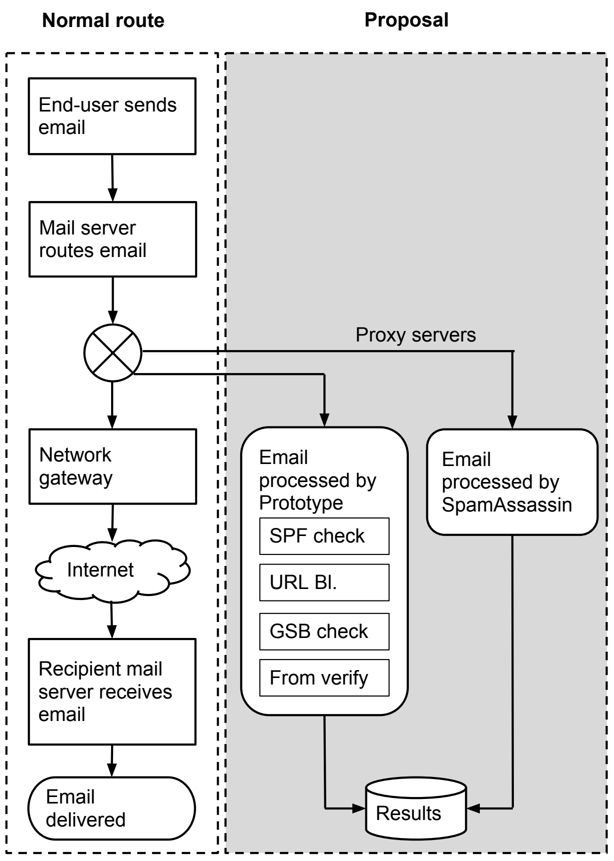

Figure 2. Email flow

the FROM check uses the *Mail::CheckUser* Perl library. All results are logged by extracting the different added email headers adding the time and the process time of the email. The experiment uses the techniques in the Prototype exactly as described by de Vries et al., the SPF and blacklist checks are performed for every email, but the FROM-address check only if there is no hit from these three checks [20, 21]. The alternative setup, for comparison, uses the *spamassassin* plugin for qpsmtpd.

To measure hardware performance a simple script was written using the Perl library *Sys::Statistics::Linux*. The script writes the system resource usage to a log file within one minute intervals. The virtual machines make use of 2 GB memory and four CPU cores. The hardware utilised by the virtual machines is new and high-end, employing Intel® Xeon® E5-2640 CPUs and 1333 MHz DDR3 Registered ECC memory.

5. RESULTS

In this section the results of the measurements associated with the case study are described. The Prototype and SpamAssassin setup are compared on effectiveness and efficiency. First the results concerning effectiveness are examined, second the efficiency of both setups is reviewed and finally the whole case study is discussed.

During the case study a constant flow of emails was processed by the proxy servers. During the nights the number of emails was significantly lower than during the day. In Figure 3 the average number of emails processed during the hours of the day is shown. These numbers are the same for both proxy servers, as they process the same emails.

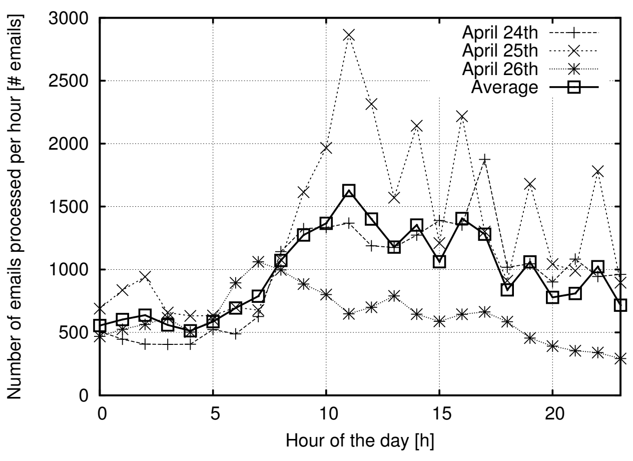

Figure 3. Average number of emails processed for each hour of the day

5.1 Effectiveness

In this section the results concerning the effectiveness performance metric are described. The detected spam emails of the Prototype are compared with the detected spam emails of the SpamAssassin setup by calculating the observed percentage by dividing the number of detected spam emails by the Prototype also marked by SpamAssassin by the total number of emails marked by SpamAssassin. These percentages are compared with the results of de Vries in Table 2. Please note that the results of de Vries were also calculated by using emails marked as spam by another source as the observed emails.

Secondly, the detected spam emails of the Prototype are also compared with the detected spam emails of the SpamAssassin setup by calculating the observed percentage dividing the number of detected spam emails by the Prototype as well as the SpamAssassin setup by the number of detected spam emails by the Prototype. The results are shown in Figure 4. The remaining part can be considered as being false positives, and thus wrongfully marked spam by the Prototype, assuming that the SpamAssassin setup has no false negatives.

The first notable result concerning the effectiveness of the Prototype is the number of emails marked by the Google Safe Browsing check. In total 13 emails, about 0.02% of all scanned email, are marked, none of which also have a SpamAssassin score high enough to be considered as spam. The GSB check will not be regarded in remainder of the results.

The measurements of de Vries et al. were performed on email marked spam. As a comparison only the emails marked as spam by the SpamAssassin setup are considered, a total of 4338 emails, 6.24% of all emails scanned. The comparative results are summarised in Table 2. If the three checks are combined, then 99.1% of the emails marked by the SpamAssassin setup is also marked by the Prototype.

Table 2. Comparison with results de Vries et al.

	SPF hit	URIBL hit	FROM hit
De Vries	38.6%	74.4%	100%
Prototype	97.0%	29.9%	41.1%

The results obtained by de Vries et al. differ greatly from the results gained from measurements in a real life environment. As every day different emails are sent and the behaviour of spammers cannot be predetermined, the differences might be caused by just that. In the Discussion section other possible explanations are considered.

In the real life environment also emails not predefined to be spam are checked using the four proposed Prototype techniques. This gives the opportunity to determine which percentage of the total email traffic is marked by the proposed Prototype techniques and more importantly how many of these emails are at the same time also marked by SpamAssassin (Figure 4).

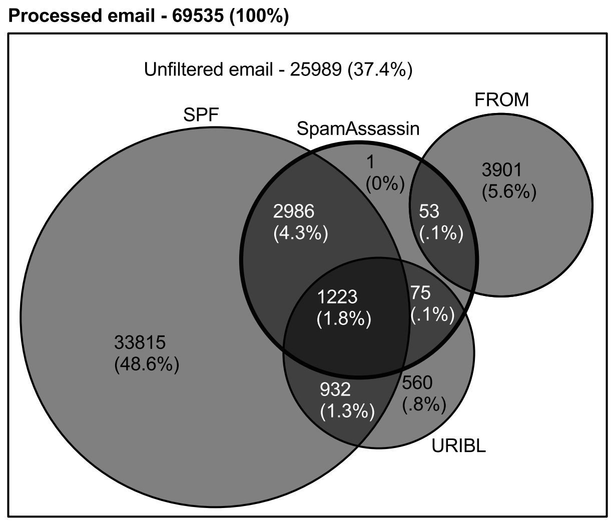

Figure 4. Results considering all processed email

Two striking observations can be made regarding the results summarised in Figure 4. First the percentage of email hit by the SPF check is very high, which means that a large portion of email scanned did not use a sender address provided by the server from which the email was sent. Secondly the emails hit by the Prototype are not marked by the SpamAssassin setup for a large percentage of the time. Only 46.5% of emails hit by the URIBL check was also marked as spam by the SpamAssassin setup which is much lower than expected, since email containing blacklisted URLs will probably be spam.

As also can be seen in Figure 4, all except one email marked by SpamAssassin are also marked by the Prototype. However a much greater part of emails marked by the Prototype are not marked by SpamAssassin. These marked emails can be considered as being false positives.

The Prototype does seem to be effective in establishing which emails are not spam. The emails containing no blacklisted URLs and passing the SPF check, 19944 (28,7%) emails in total, were in 99.7% of the cases also not marked by the SpamAssassin.

5.2 Efficiency

In this section the results concerning the efficiency performance metric are described. Effciency is a relative metric calculated by measuring the different aspects of hardware utilisation of the techniques. In particular, an efficiency comparison is made between the Prototype and SpamAssassin setup based on "average email processing time" and the "average CPU utilisation". The email processing time is determined by measuring the number of microseconds between the start of an SMTP transaction from the HELO/EHLO command until the point the email is queued for further delivery. During this processing time an email processed by the Prototype uses all four spam stopping techniques. An email that is processed by SpamAssassin uses only the spam stopping technique that is incorporated in the SpamAssassin implementation.

The CPU utilisation is measured automatically every minute and is determined by combining the CPU utilisation of the user and the system.

The efficiency of the two setups is measured by the process times per email and the impact on the hardware usage. As the number of emails the proxy servers should be able to process is high, the email processing times should be as low as possible. The average process times together with their 95% confidence intervals are shown in Figure 5.

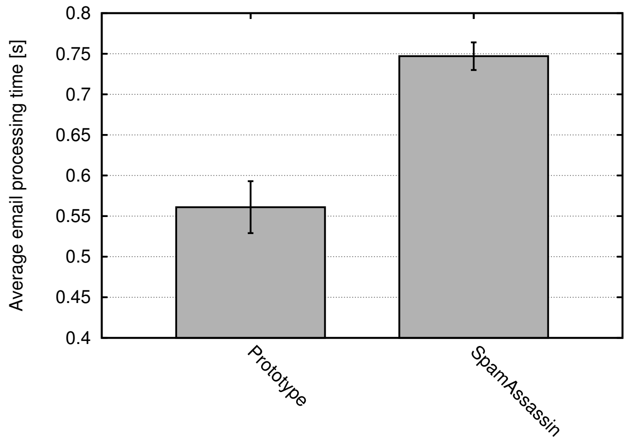

Figure 5. Average time to process an email with 95% confidence intervals

Figure 5 clearly shows that the average processing time of emails is lower for the Prototype than for the SpamAssassin setup. When the averages and confidence intervals are calculated per day or per hour at some points the confidence intervals overlap and for a few hours the calculated average email processing time is even higher for the Prototype. At hour 19 of april 26th a large peak for the average process time of emails is observed for the Prototype. A possible reason for these higher processing times is a temporarily slower DNS service. As three of the four techniques used at some point need to perform a DNS request, a slower DNS service has a great impact on the observed processing times.

To determine the hardware usage the average CPU utilisation of both setups are presented in Figure 6. In contrast to the processing times the SpamAssassin setup performs better than the Prototype when analysing the hardware usage. The SpamAssassin setup has a lower CPU utilisation average. However, even more so than for the average processing time the confidence intervals overlap when the averages are calculated on a per day or on a per hour basis.

5.3 Discussion

5.3.1 *Effectiveness*

The blacklist checks, especially the Google Safe Browsing check, performed worst when the Prototype was used compared to the measurements performed by de Vries et al.. This can possibly be explained by the delay between sending the email and performing the checks in the original (by de Vries et al.) experiment. As the blacklists are constructed by reports about URLs already used in spam emails or by just scanning Internet pages periodically. The URL in spam emails may not yet be listed when the email is sent, but it might be listed soon after that.

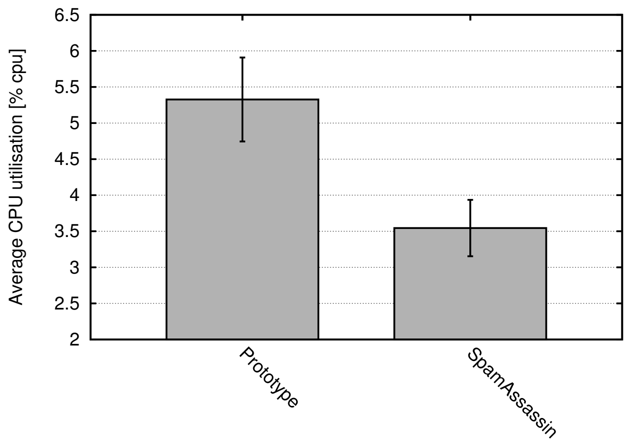

Figure 6. Average CPU utilisation with 95% confidence intervals

The SPF check performed significantly better when the Prototype is used compared to the results shown by de Vries et al.. As the SPF check can return several states, the difference in performance might be caused by a different interpretation of the returned states. The different states are described in RFC 4408 [22]. In the results presented in this paper only *pass* was considered as a positive result obtained from the SPF check. As the mail servers should only be used by domain names containing the mail server's IP-address in the SPF record. This is correctly set for all domain names within the cooperating ISP's network.

A large percentage of all emails processed does not pass the SPF check in the Prototype, and just over 10% of these emails are also considered spam by SpamAssassin. Based on this fact it can be concluded that many of the ISPs end-users do not send email correctly. For example some end-users use a GMail address in an email that is sent using the ISPs mail servers. The SPF check will return a *softfail* state, as the email is not sent correctly, but the email itself might still be legitimate.

The URIBL check was only in 46.5% of processed emails also marked by the SpamAssassin setup. This is much lower than expected. SpamAssassin by default only assigned 3 points to blacklisted URLs, while 5 points are necessary to mark an email as spam. Furthermore for example the SPF check performed by SpamAssassin can have a positive influence on the total score when the result is pass. The only way to discover what type of emails might contain blacklisted URLs but are not spam emails is to analyse the emails manually.

Emails passing the SPF check and not containing blacklisted URLs are not marked as spam for 99.7% of processed email. The SPF and blacklist checks mainly consist of DNS checks. As these checks are fast and effective, see email processing time results, they can be used to efficiently stop a large portion of spam, without having to perform further processing. However, additional spam stopping checks can be done by i.e., SpamAssassin on the remaining emails. An advantage of SpamAssassin is that emails are not marked as spam but are scored. The ISPs can determine the spam stopping threshold by themselves.

5.3.2 *Efficiency*

The Prototype did not outperform the SpamAssassin setup concerning the hardware load. This may be caused by the FROM address check. The check is not performed for every email, but it is a relatively intensive check, since an SMTP transaction has to be setup. This is the only check that needs the use of a TCP connection and multiple messages. This causes an increase in the processing time and as a consequence the load on the system also increases. The CPU utilisation of the SpamAssassin setup was lower than expected. SpamAssassin was thought to be resource intensive since the probability calculations use a relatively large amount of CPU cycles. It seems however that SpamAssassin has a low impact on the performance of the currently available hardware.

The email processing time as well as the CPU utilisation measurements show high fluctuation and as a result the confidence intervals for the calculated averages are large when determined per hour of day. To determine which email spam stopping setup (Prototype or SpamAssassin) performs better from the point of efficiency, additional measurements for multiple days are needed.

6. CONCLUSIONS AND FUTURE WORK

In this paper the work of de Vries et al. is further explored and the described techniques are tested in a real life environment. De Vries et al. developed lightweight techniques to stop spam at the sender side. From the measurements performed and analysed in this paper it can be concluded that, when compared to the spam stopping scores provided by SpamAssassin, the spam stopping techniques incorporated in the Prototype solution cannot be considered as being reliable in marking spam emails. Too many emails would be incorrectly marked that can be regarded as false positives.

The proposed spam stopping techniques were effective in determining which emails are unlikely to be a spam email. These techniques can thus be used as a first spam stopping filter, while other spam stopping solutions, e.g., SpamAssassin can be used to filter the remaining emails.

Assessing the hardware used and the email throughput during the experiment both setups can be considered lightweight. However, in the accomplished experiments the spam stopping techniques incorporated in the Prototype did not outperform the SpamAssassin setup from the point of efficiency. As a consequence these techniques cannot be classified as being more lightweight than SpamAssassin.

In summary, the Prototype techniques by themselves cannot effectively and efficiently stop spam at the sender side. However, they can be very well be used in combination with SpamAssassin.

As the techniques used in the Prototype proved to be effective in finding out legitimate emails, the efficiency of a setup using the SPF and blacklist check as a first spam stopping filter and SpamAssassin as a second one can be studied. The effectiveness should be the same for both setups as measurements suggest. A similar experiment can be performed excluding the FROM address check, in order to determine whether this techniques is the main cause of increasing the system load.

Improvements to this research can be made by gathering more data. Due to time restrictions a limited number of measurements could be performed. In order to eliminate the SpamAssassin false negatives, a manual verification of the emails could be performed.

Further research is also recommended in determining the type of emails that are containing blacklisted URLs and are not marked as spam by SpamAssassin, since emails containing blacklisted URLs should be considered as being spam emails.

7. ACKNOWLEDGMENTS

The author would like to thank the members of the Design and Analysis of Communication Systems (DACS) group at the University of Twente for valuable feedback and advice, especially Georgios Karagiannis Ph.D. and Jair Santanna M.Sc. Also the author would like to thank Antagonist B.V for providing the hardware and environment needed for the experiments.

8. REFERENCES

[1] Backscatterer.org. Sender callouts - why it is abusive. `http://www.backscatterer.org/?target=sendercallouts`, 2013. (Retrieved on: 25/04/13).

[2] G. Caruana and M. Li. A survey of emerging approaches to spam filtering. *ACM Computing Surveys*, 44(2):1–27, Feb. 2012.

[3] R. Clayton. Stopping spam by extrusion detection. In *First Conference on Email and Anti-Spam (CEAS 2004), Mountain View CA, USA*, pages 30–31, July 2004.

[4] R. Clayton. Stopping outgoing spam by examining incoming server logs. In *Second Conference on Email and Anti-Spam (CEAS 2005), Stanford, CA*, July 2005.

[5] Commtouch. Solutions for Service Providers. `http://www.commtouch.com/solutions-service-providers/`, 2013. (Retrieved on: 11/04/13).

[6] D. Cook, J. Hartnett, K. Manderson, and J. Scanlan. Catching spam before it arrives: domain specific dynamic blacklists. In *ACSW Frontiers '06: Proceedings of the 2006 Australasian workshops on Grid computing and e-research.* Australian Computer Society, Inc, Jan. 2006.

[7] J. E. Dunn. ISPs reluctant to filter outbound spam. `http://www.networkworld.com/news/2010/061010-isps-reluctant-to-filter-outbound.html`, June 2010. (Retrieved on: 25/02/13).

[8] Google. Safe Browsing API. `https://developers.google.com/safe-browsing/`, Apr. 2012. (Retrieved on: 07/03/13).

[9] D. Gudkova. Kaspersky Security Bulletin: Spam Evolution 2012. `http://www.securelist.com/en/analysis/204792276/Kaspersky_Security_Bulletin_Spam_Evolution_2012`, Jan. 2013. (Retrieved on: 18/02/13).

[10] R. Jain. *The art of computer systems performance analysis.* John Wiley & Sons, New York, Apr. 1991.

[11] J. C. Klensin. Simple mail transfer protocol. RFC 2821, IETF, April 2001.

[12] MailChannels. The most powerful outbound anti-spam. `http://www.mailchannels.com/`, 2013. (Retrieved on: 11/04/13).

[13] C. McGregor. Controlling spam with SpamAssassin. *Linux Journal*, 2007(153), Jan. 2007.

[14] qpsmtpd. qpsmtpd. `http://smtpd.develooper.com`, 2013. (Retrieved on: 07/03/13).

[15] F. Ridzuan, V. Potdar, and A. Talevski. Factors involved in estimating cost of email spam. In *ICCSA'10: Proceedings of the 2010 international conference on Computational Science and Its Applications.* Springer-Verlag, Mar. 2010.

[16] Spamassassin. Welcome to SpamAssassin. `http://spamassassin.apache.org/`, 2013. (Retrieved on: 07/03/13).

[17] A. Sperotto, G. Vliek, R. Sadre, and A. Pras. Detecting spam at the network level. *The Internet of the Future*, pages 208–216, 2009.

[18] SURBL. SURBL. `http://www.surbl.org/`, 2012. (Retrieved on: 07/03/13).

[19] URIBL. Realtime URI Blacklist. `http://www.uribl.com/`, 2012. (Retrieved on: 07/03/13).

[20] W. W. de Vries. Restraining transmission of unsolicited bulk e-mail. *Proceedings of the twelth Twente Student Conference on Information Technology*, Jan. 2010.

[21] W. W. de Vries, G. C. M. Moura, and A. Pras. Fighting spam on the sender side: a lightweight approach. In *EUNICE'10: Proceedings of the 16th EUNICE/IFIP WG 6.6 conference on Networked services and applications: engineering, control and management*. Springer-Verlag, June 2010.

[22] M. Wong and W. Schlitt. Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. RFC 4408 (Experimental), IETF, April 2006.