An Anti-Spam Method with SMTP Session Abort

Tempfailing and distributed collaborative spam filtering are typical anti-spam technologies commonly used in many organizations. However, these technologies have some drawbacks in terms of operation or performance. In this paper, in order to reduce the drawbacks, we propose an anti-spam method introducing SMTP session abort function, collaborating with these two technologies. We also show the effectiveness of the proposed method by means of operation tests.

1 Introduction

E-mail is one of the most popular services on the Internet as an essential communication medium for social activities today. However, proliferation of spam mails is a serious issue of the Internet.

There are several anti-spam technologies commonly used in many organizations, such as blocking, filtering, and so on.

Blocking is an anti-spam technology that works at the start of or during an SMTP session, based on the sender address for example. One typical blocking method is tempfailing, which temporarily refuses the first delivery attempt of a message from an untrusted Mail Transfer Agent (MTA). This method can refuse spam mails considerably since few spam senders retry to send the temporarily failed message. However, it also may refuse legitimate mails sent from such domains that resend the temporarily failed message with a different (fallback) MTA. In such a case, an administrator of the receiver MTA has to register those domains by hand so that the receiver MTA accepts any messages from them at the first time.

On the other hand, filtering is another anti-spam technology that works after receiving a message, based on its contents such as its header and body. One typical filtering method is distributed collaborative spam filter (also known as signature based filter), which refuses the same messages that some recipients register in a distributed database as a spam. This method reduces false positive rate considerably. However, it requires a huge number of users to collaborate for good performance.

In this paper, we propose an anti-spam method for reducing the above problems. This method performs the same effect as existing tempfailing methods by means of SMTP session abort during the first delivery attempt. In addition, this method can obtain the header and body of a message even if it would not be resent and can use the header for second delivery checking or the body of unresent messages for registering to the database for example.

2 Existing anti-spam methods

As described in the previous section, many anti-spam methods have been proposed so far. In this section, we describe overviews and problems of two typical methods relating to the proposed method.
2.1 Tempfailing

According to RFC2821[1], if a sender-SMTP agent receives a response “4yz”, which indicates a temporary error, from a receiver-SMTP agent, the sender-SMTP agent should retry to send the temporarily failed message after waiting for a specified period. However, most spam sender MTAs violate this rule since they generally prefer throughput to reliability of spam mail delivery. Tempfailing method distinguishes spam mail senders from legitimate mail senders in terms of the difference of these behaviors. Specifically, when a sender-SMTP agent starts an SMTP session to a receiver-SMTP agent with tempfailing function, the receiver-SMTP agent decides at first whether it is the first delivery attempt or not. If so, the receiver-SMTP agent replies a temporary error intentionally, looking forward to receiving the same message again. Otherwise, the receiver-SMTP agent just accepts the message as usual.

One typical tempfailing method greylisting[2, 3] uses a triplet (sender IP, SMTP From, SMTP To) to decide whether an SMTP session is for the first delivery attempt or not. This method also uses a database called a whitelist that contains trusted sender-SMTP agents. If a sender-SMTP agent in the whitelist establishes an SMTP session, the receiver-SMTP agent accepts this session without replying a temporary error, regardless of whether it is the first delivery attempt or not. Usually, a sender-SMTP agent that retries to send the temporarily failed message is automatically registered in the whitelist.

Tempfailing works effectively in spite of its simple mechanism. However, this method causes large delay of legitimate mail delivery as follows. Since RFC2821 recommends 30 minutes at least as the retry interval, legitimate mail delivery from a sender-SMTP agent not registered in the whitelist may be delayed as much. In addition, if the second time session is established from a different agent (as known as a fallback MX host) as seen in large domains, the receiver-SMTP agent may decide it as another first delivery attempt. Accordingly, the delay of legitimate mail delivery becomes larger. To resolve this problem, an administrator of the receiver-SMTP agent has to register the first sender-SMTP agent in its whitelist manually.

Furthermore, this method has another problem that it has no recovery process after a false positive occurs. That is, if a sender-SMTP agent fails to resend a legitimate message to a receiver-SMTP agent with tempfailing function, the receiver-SMTP agent may reply a temporary error and the message will be lost. To resolve this problem, an administrator of the receiver-SMTP agent has to register such sender-SMTP agents in its whitelist as well. However, since only the triplet (sender IP, SMTP From, SMTP To) is recorded in the receiver-SMTP agent, it is difficult for the administrator to decide whether the message was legitimate or not.

2.2 Distributed collaborative spam filter

Distributed collaborative spam filter is a kind of anti-spam method based on the fact that spam senders typically send the identical messages to many recipients. This method introduces a spam mail database shared by all recipients to detect spam messages. Specifically, each recipient cooperates with others in registering his/her spam mails in the database and filters out spam mails according to whether the message is registered in the database or not. For efficiency, this database typically contains a kind of checksum called a signature of each spam mail instead of the whole message. Distributed Checksum Clearinghouse (DCC)[4], Razor[5], Pyzor[6] are typical implementations of this method.

This method is superior to other methods in terms of negligible false positive rate since spam mails are registered to the database according to judgment by recipients. However, it needs a great number of spam mails registered in the database early to improve false negative rate.

To register many spam mails into the database effectively, some researchers have introduced a kind of honeypots deployed over the Internet[7, 8]. However, since distribution of spam mails has some locality, honeypots running on other domains or hosts could not collect spam mails relevant to the target domain so effectively.

3 Anti-spam method with SMTP session abort

As mentioned in the previous section, existing anti-spam methods, namely tempfailing and distributed collaborative spam filter have some problems on their operation or performance. In this section, in order to resolve these problems, we propose an anti-spam method based on combination of both methods, by virtue of SMTP session abort.

3.1 Overview of the proposed method

In the proposed method, in order to implement both tempfailing and distributed collaborative spam filter without modifying existing MTAs, we introduce mail gateways with both functions. A sample of system layout is shown in Figure 1. In this figure, there ex-
Figure 1: A system layout of the proposed method

There are two mail gateways, which are designated as the primary MX (PMG) and the secondary MX (SMG), respectively, of MTAs depicted at the right side by DNS setting. A database is also introduced for distributed collaborative spam filter. Note that a single mail gateway may be designated as both the primary and secondary MXes, as shown in Section 5.2.

Each mail gateway has SMTP session abort function by means of sending a packet setting RST flag on. During the first delivery attempt, the mail gateway aborts this session after receiving the header or the whole message, according to the system configuration. Therefore, if an SMTP client wants to send a message to one of the MTAs in Figure 1, the SMTP client at first tries to send the message to PMG according to the primary MX, and it fails due to session abort. Then the SMTP client immediately tries to send the message to SMG according to the secondary MX and SMG accepts this message. Thus, introduction of multiple mail gateways reduces delivery delay on existing tempfailing method.

3.2 Improvement of processing for retransmission judgment

As mentioned in the previous section, existing tempfailing methods such as greylisting have a problem that they cannot accept the second time delivery from a different SMTP client since the sender IP address is used for retransmission judgment. Therefore, instead of the sender IP address, the proposed method uses some header fields such as “Message-ID:” or the message body (or its checksum) for retransmission judgment. When PMG accepts the first delivery attempt from an untrusted SMTP client, PMG obtains the header or the whole message and then aborts this session. After that, when the SMTP client retries to send the same message to SMG, SMG obtains the header or the whole message and identifies that the message being received is the same one as PMG aborted to receive, based on the triplet (Message-ID, SMTP From, SMTP To) for example. Thus, the proposed method accepts a message from an untrusted SMTP client even in case of the second time delivery from a different SMTP client. Note that according to RFC2822[9], “Message-ID:” field is not mandatory. We will discuss the case where “Message-ID:” field is not found in a message in Section 5.3.

3.3 Automatic registration of spam mails

In existing distributed collaborative spam filter, only spam mails destined to valid recipients are to be registered to the spam database but those destined to non-existing recipients are simply discarded. Although mails to non-existing recipients, especially those not resent in case of tempfailing, are almost always spam mails, they are not registered into the spam database.

In the proposed method, since we can obtain the whole messages destined to non-existing recipients even if they are not resent after tempfailing, many potential spam mails are to be registered into the spam database. In practice, the proposed method registers each message destined to non-existing recipients at the first delivery attempt and cancels the previous registration at the second time delivery if exists. This process would improve the performance of distributed collaborative spam filter more than existing honeypots since it collect spam mails specific to the same domain early and effectively. Note that the proposed method can also register the unresent messages sent even to existing recipients. However, we think this automatic registration is subject to the recipients’ approval for ethical or privacy reason.

3.4 Overall procedure

In this subsection, we show the typical overall procedure of the proposed method in detail. Note that the following procedure describes only the activity of PMG. However, that of SMG is the identical since there exist many MTAs (especially spam sender MTAs) that ignore the preference of MX records.

1. A sender-SMTP agent establishes an SMTP session to the primary MX host, which is designated to PMG. PMG keeps track of the IP address of the sender-SMTP agent. If this IP address is registered in the whitelist maintained by the system administrator, PMG deals with this SMTP session as usual and terminates the procedure.

2. The sender SMTP agent issues “MAIL” and
“RCPT” commands to PMG, specifying the sender address and the recipient address, respectively. Then PMG returns a positive response “250 OK” to the sender MTA for “RCPT” command, regardless of the recipient address. PMG also keeps track of the sender address and the recipient addresses for later first delivery attempt check.

3. The sender-SMTP agent issues “DATA” command. PGM accepts this command and proceeds to receive the body of the message.

4. While receiving the body of the message, PMG retrieves the Message-ID in the header, and checks whether this session is for the first delivery attempt or not, based on the triplet (Message-ID, SMTP From, SMTP To) for example.

5. If this session is for the first delivery attempt, PMG aborts it after receiving the header or the whole body of the message according to the system configuration, in expectation of the second delivery attempt. Then PMG registers the header or the whole body of the message into the spam database and terminates the procedure. If only the messages destined to non-existing recipients should be registered into the spam database, PMG checks the non-existence of the recipient addresses before the registration.

6. Otherwise, this session is for the second delivery attempt. PMG deals with this SMTP session as usual and cancels (decrements the registration count of) the previous registration of the spam database.

3.5 User preference of abort timing

In the proposed method, network traffic and delivery delay depend on whether session abort is performed or not and when if it is performed. We therefore introduce a user preference function on session abort timing. Options are “accept” which means no session abort, “header” which means session abort after the end of the header, and “body” which means session abort after the end of the whole message.

With this function, a new problem arises if multiple recipients are given. That is, if options selected by recipient users are different, what should the receiver MTA do? Table 1 shows the action of the receiver MTA, namely when the SMTP session is aborted, whether the message is delivered to the users specifying “accept” option or not, and whether recipient addresses are modified or not, on the first delivery attempt to work around this problem. If the message is destined to both users with “accept” option and other users as shown as #3, 5, 7 in Table 1, it is delivered to the former users on the first delivery attempt and to the latter users on the second delivery attempt.

4 Implementation of the prototype system

According to the system layout and the procedure shown in the previous section, we implemented a prototype system for the proposed method. In the prototype system, we introduced sendmail to both PMG and SMG running FreeBSD. We also introduced DCC as a spam database, which contains the checksum of the whole body. PMG and SMG were configured to abort SMTP sessions after receiving the whole message. SMTP session abort function was implemented as an external program, which creates and sends a packet with RST flag to both sender- and receiver-SMTP agents. As a method of second delivery checking, we adopted the triplet (Message-ID, SMTP From, SMTP To), as mentioned above. As for registration into the spam database, only the messages destined but not resent to one or more non-existing recipients were registered since we could not have obtained approval of the all potential recipients in advance.

5 Preliminary results and consideration

5.1 First operation test

In order to verify the behavior of the prototype system, we firstly examined its behavior on an experimental network environment isolated from the Internet. According to the results, we confirmed that all the following functions work well: SMTP session abort function, second delivery attempt check function, database registration/cancellation function, and user preference function.

Secondly, we carried out an operation test of the prototype system on the Internet environment. In this test, the system was configured to deal with all messages destined to some domains of Okayama University that were to be obsolete and to forward them to the original receiver-MTAs, by means of replacement of MX records. Most of the messages processed by the prototype system were destined to non-existing recipients and most of them would probably be spam mails. However, we were not sure how many spam mails were processed since we could not verify the contents of the mail processed for privacy reason. This operation test was carried out for seven days from January 29th to February 5th, 2006.

During the operation test, in order to evaluate the
Table 1: Action with multiple recipients.

<table>
<thead>
<tr>
<th>#</th>
<th>User preferences</th>
<th>abort</th>
<th>message delivery</th>
<th>recipients</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>accept</td>
<td>header</td>
<td>body</td>
<td>accept</td>
</tr>
<tr>
<td>1</td>
<td>√</td>
<td>header</td>
<td>body</td>
<td>YES</td>
</tr>
<tr>
<td>2</td>
<td>header</td>
<td>NO</td>
<td>retained</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>√</td>
<td>√</td>
<td>body</td>
<td>YES</td>
</tr>
<tr>
<td>4</td>
<td>body</td>
<td>NO</td>
<td>retained</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>body</td>
<td>YES</td>
<td>modified</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>body</td>
<td>NO</td>
<td>retained</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>body</td>
<td>YES</td>
<td>modified</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: The result of the first test operation.

<table>
<thead>
<tr>
<th></th>
<th>#resent</th>
<th>#not resent</th>
<th>#processed</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMG</td>
<td>5,340</td>
<td>34,415</td>
<td>39,755</td>
</tr>
<tr>
<td>SMG</td>
<td>5,076</td>
<td>9,888</td>
<td>14,964</td>
</tr>
<tr>
<td>total</td>
<td>10,416</td>
<td>44,303</td>
<td>54,719</td>
</tr>
</tbody>
</table>

effectiveness of the blocking function, we analyzed the log of the system and counted the number of mails resent and that of mails unresent, after SMTP session abort. The result is shown in Table 2.

According to the result shown in this table, the prototype system blocked 44,303 of possible spam mails, 81% of 54,719 mails processed. This blocking performance was as well as that of greylisting. Consequently, we can say that SMTP session abort function works well in terms of a message blocking method.

Since there still existed many spam mails in 10,416 mails resent after SMTP session abort, some of them might be registered to the spam database already and be filtered out. In order to evaluate distributed collaborative spam filtering function, we did further analysis. According to the analysis result, we counted 2,180 mails filtered out, which is 20% of all the mails resent. Note that we could not obtain the exact filtering rate since the number of legitimate mails was unknown for privacy reason. This rate seems relatively high and would be larger in practice, considering the fact that the spam database had only unresent mails destined to non-existing recipients during the test period of seven days.

5.2 Second operation test

Since we could not compare the proposed method with existing temperfailing method in the first operation test, we did the second operation test in January and February 2008. This test was carried out on a new domain since the domain used in the previous test had already been obsolete. In this test, we could not evaluate false negative rate nor false positive rate since there was no usual mail traffic on this new domain. Instead, we utilized some free mail services, mailing lists and personal mails from ISPs, and investigated activities of retry function (with or without a different MTA) and the minimum retry interval. The names of MTA software used were also investigated if they were recorded in Received fields.

In the second operation test, we utilized a mail gateway as both PMG and SMG for easy administration since many small companies at least in Japan only use one IP address. Therefore, we set DNS resource records like the followings:

$ORIGIN example.com.
@ IN MX 10 pmg
@ IN MX 20 smg
pmg IN A 192.0.2.123
smg IN A 192.0.2.123

The results for some domains are shown in Table 3. As shown in this table, all domains investigated had retry function and a few domains such as gmail.com utilized fallback MX function. Consequently, we verified that the proposed method could accept all messages from all domains investigated without any whitelist registration, although conventional greylisting could not accept the resent messages properly.

As for retry interval, many domains retried to send the temporarily failed message within about 10 seconds. This fact implies the proposed method reasonably reduces the delay of legitimate mail delivery mentioned in the previous section. Some domains using qmail still had large delay. This was because qmail was designed to resend messages to the same (primary) MX to which it established an SMTP session and then it failed. The delay of gmail.com would also be explained by the same reason.

There were still many domains that had large delay other than those in Table 3. Another possible reason of those delays is the following. Since PMG and SMG had the same IP address in this test, they would treat
Table 3: The result of the second test operation.

<table>
<thead>
<tr>
<th>domain (service)</th>
<th>MTA software</th>
<th>resend</th>
<th>different MTA</th>
<th>min. interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>cc.okayama-u.ac.jp (university)</td>
<td>sendmail</td>
<td>YES</td>
<td>NO</td>
<td>0 (sec)</td>
</tr>
<tr>
<td>nifty.com (ISP)</td>
<td>sendmail</td>
<td>YES</td>
<td>NO</td>
<td>1</td>
</tr>
<tr>
<td>listbox.com (spf-discuss ML)</td>
<td>postfix</td>
<td>YES</td>
<td>NO</td>
<td>1</td>
</tr>
<tr>
<td>yahoo.com (free mail)</td>
<td>?</td>
<td>YES</td>
<td>YES</td>
<td>385</td>
</tr>
<tr>
<td>gmail.com (free mail)</td>
<td>?</td>
<td>YES</td>
<td>YES</td>
<td>10</td>
</tr>
<tr>
<td>aol.com (free mail)</td>
<td>?</td>
<td>YES</td>
<td>NO</td>
<td>6</td>
</tr>
<tr>
<td>hotmail.com (free mail)</td>
<td>SMTPSVC</td>
<td>YES</td>
<td>NO</td>
<td>6</td>
</tr>
<tr>
<td>yahoogroups.jp (free ML)</td>
<td>?</td>
<td>YES</td>
<td>NO</td>
<td>1</td>
</tr>
<tr>
<td>freeml.com (free ML)</td>
<td>qmail</td>
<td>YES</td>
<td>NO</td>
<td>399</td>
</tr>
<tr>
<td>mag2.com (mail magazine)</td>
<td>qmail</td>
<td>YES</td>
<td>NO</td>
<td>3264</td>
</tr>
<tr>
<td>trashmail.net (anonymous mail)</td>
<td>postfix</td>
<td>YES</td>
<td>NO</td>
<td>6</td>
</tr>
</tbody>
</table>

5.3 Discussion about false positives

We did not find any false positives in the second operation test. However, a few false positives can happen in actual operation for various reasons. We discuss possible false positives and their countermeasures in this section.

5.3.1 Messages without Message-ID

As described in Section 3.2, RFC2822 does not always require “Message-ID:” field in the message. Therefore, retransmission judgment based on a triplet (Message-ID, SMTP From, SMTP To) may fail to work well. Actually, some messages from yahoo.com or yahoogroups.jp did not have “Message-ID:” field.

One possible solution against this problem is to utilize either the checksum of the body or “Date:” field, which is mandatory according to RFC2822, instead of Message-ID. The former can be used for more precise retransmission judgment but it is not applicable for session abort after the end of the header. On the other hand, the latter is always applicable but it may receive a message on the first delivery attempt if two messages are sent from the same MTA within one second since the granularity of “Date:” field is a second.

In the prototype system used in the second operation test, we implemented retransmission judgment function with “Date:” field. Consequently, it received all messages without Message-ID including those from yahoo.com or yahoogroups.jp.

5.3.2 MTAs without retransmission

As shown in [3], there exist some legitimate MTAs that do not retry to send although they were not found in the second operation test. With the existing tempfailing methods, messages from such MTAs are to be lost. In order to work around this problem, the user or the administrator of the receiver MTA should register such sender MTAs into the whitelist. However, since only the triplet (Sender IP, SMTP From, SMTP To) would be logged at most, it is difficult for the user or the administrator to decide whether the unresent message is legitimate or not.

The proposed method is also to lose messages from such sender MTAs unless they are registered into the whitelist. However, this method can save the header or the whole messages rather than the triplet mentioned above, by virtue of session abort after the end of the message. Therefore, it is much easier than the existing tempfailing methods for the user or the administrator to decide whether the unresent message is legitimate or not.

Note that most Mail User Agents (MUAs) do not retry to send nor put a Message-ID to the message. However, since messages submitted by MUAs should be dealt with Mail Submission Agents (MSAs) rather than MTAs nowadays, this would not be a problem in practice.

5.3.3 MTAs changing SMTP From address

There also exist some legitimate MTAs that change SMTP From address on the second delivery attempt although they were not found in the second operation test. According to [3], Bounce Address Tag Validation (BATV)[10] is a technique that may change the SMTP From address since it puts a time stamp in that address. Both of the existing tempfailing methods and
the proposed method are to fail to accept messages with a different SMTP From address.

One possible technique to work around this problem is to utilize only a pair (Message-ID, SMTP To) without including SMTP From since SMTP From is redundant information to identify a message if a Message-ID is supplied. However, this technique would not work well if both SMTP From and Message-ID are changed for every delivery attempt. In this case, only solution against this problem is to use the whitelist.

6 Conclusions

In this paper, we proposed an anti-spam method introducing SMTP session abort function, collaborating with tempfailing and distributed collaborative spam database. This method reduces the drawback of existing methods. We also implemented a prototype system and confirmed the effectiveness of the proposed method by means of operation tests.

Further works include performance evaluation of the proposed method through actual long term operation.

Acknowledgments

This research was partially supported by Japan Society for the Promotion of Science (JSPS), Grant-in-Aid for Scientific Research (B) No. 17300038 in 2005–2007.

References


