# Two-Way Mail

A proposal to change Internet Email to use a bidirectionally authorized transmission protocol.

Version 0.2 – 2015-02-25 - initial draft

Kai Engert, Red Hat Inc.
kaie@kuix.de and kaie@redhat.com

Check https://kuix.de/two-way-mail/ for updates to this document.

## Summary:

Email should adopt the both-way opt-in requirement used by many instant messaging systems, before messages are accepted for delivery.
Require email address whitelists, controlled by the user, managed on the user's mailbox server.
Find a way to signal contact attempts to the recipient, enabling a user to discover them and add entries to the whitelist.
Change email delivery from today's store-and-forward approach and switch to store-notify-and-poll.
Have the destination server retrieve email only if the sender address is in the recipient's whitelist.
Have the destination server contact a mail server based on official records (e.g. DNS), instead of accepting mail from anywhere.

## Motivation

Today, email can be sent to any email address without authorization from the recipient. This enables the sending of unsolicited email, also known as spam. Although there have been many attempts to eliminate spam, we still live with the difficicult situation that we can miss email because of incorrectly filtered email, and we still receive spam emails because automatic detection mechanisms are insufficient. In addition, because confirming the receipt of email can encourage senders to send more spam, today senders of Internet email don't know if a message has reached the recipient, or if it has been filtered out.

This document has suggestions to enhance Internet email to solve these problems, by changing the way email is transmitted from senders to recipients.

Enhancing email can be seen as a reasonable idea, because email is omnipresent. Owning an email account is a requirement to use a majority of personalized internet services. There hasn't been any alternative to email that has been universally adopted as email.

Today's attempts to fight spam (unwanted emails) are insufficient. Filtering causes false positives and false negatives. Email providers might block sender IP addresses, and it's difficult to notice in the first place that email might get silently discarded, and also difficult to get unblocked.

Suggestion

As of today, email is a push protocol. Anyone can send to anyone else. If an email passes technical consistency checks, email is usually accepted by the destination server and is delivered into the recipient's mailbox, even if the recipient isn't interested in receiving email from the sender.

The configuration of each destination mailbox should be extended to store a whitelist of permitted senders (a principle which is already being used in other communication networks), and allow the user to edit the whitelist.

This has the benefit of reducing spam, at the cost of making communication slightly more difficult, and requiring an adjusted solution for general purpose contact email addresses like contact@example-corporation.com.

Let's examine the various uses of email, and how this approach could work.

When two people meet and would like to exchange mail, they exchange their email addresses. It will be necessary that both persons use their email interface to configure the list of acceptable senders (the whitelist), and add the other person's email.

The same needs to happen when signing up for an Internet service. As part of registering with a web service (e.g. booking-site), in addition to providing the personal email address, an additional setup will be required. The web site needs to display the sender email address that it will use to communicate with the user (e.g. info@booking-site). The user needs to add that address to her whitelist.

At the mail transport level (the transmission of emails behind the scenes), we should move away from today's store-and-forward, and transition to a store-notify-and-poll approach.

Today, we usually have (at least) 4 entities involved when sending and receiving an email:
- the sender (S)
- the sender's email SMTP server (STS)
- the recipient's email SMTP server (RTS)
- the recipients's mailbox (R)

Today's flow is:
- S contacts and authorizes with STS
- STS temporarily stores the message and queues it for processing
- STS contacts RTS as soon as possible
- if RTS allows the message, it stores it into R
- after STS was able to deliver the message to RTS, it forgets the message

(In this description we ignore that S might do separate archiving of sent messages.)

RTS, or any filtering service that acts on behalf of RTS, might silently delete the message, and neither S nor R might ever notice.

This is a suggestion to use a different flow for email transmission, and introduce the “list of most recent contact attempts” (RCA):
- S contacts and authorizes with STS
- STS stores the message, indexed by {sender address, recipient address}
- STS creates a minimal notification for R, and queues it for processing
- STS contacts RTS as soon as possible to deliver the notification
- after STS was able to deliver the notification, it deletes the requirement for notification, but continues to store the original message
- RTS looks at R's whitelist. If the sender address S is contained in the whitelist, it queues a job to retrieve the message
- If sender S isn't in the whitelist, then RTS adds S to R's list of most recent contact attempts (RCA)
- RTS periodically processes queued jobs to retrieve messages. It will look up the appropriate DNS records to learn about the legitimate servers that are permitted to send email of the domain used by S. It will contact the server. For the outgoing connection from RTS to STS, the RTS system must use an IP that is configured in DNS as a legitimate receiving server for R's domain (potentially require authentication with a SSL certificate). STS will accept the connection, and RTS sends a request to obtain email for the pair {sender,recipient}. STS checks that the connection originates from a legitimate receiver host, or that STS has authenticated correctly. If the check passes, it will lookup the stored mail, deliver it to RTS, and delete it locally.

An STS system decides how long it is willing to store mail that is never fetched by the RTS. When it gives up, it can notify the sender by storing that information in the mailbox owned by S.

Because the STS system knows whether the recipient is willing to accept email from S, it can notify S once the message has been delivered to R.

As part of this new approach to email transmission, it is recommend that email recipients periodically check the list of recent contact attempts (RCA). The user can look at RCA regularly (e.g. once a day) whenever she is expecting an important email, and discover the need to add the missing sender to the whitelist.

It needs to be decided, if the RCA should contain only the sender's email addresses and timestamp of the sent message, or if it should also include the sender's name, and potentially even the subject of the message. It might make sense to include this information in the notification message sent from STS to RTS, and allow the recipient to configure the display of the RCA in their email interface.

At the time a new permitted sender is added to the whitelist, the RTS should contact the associated STS and retrieve any pending messages.

In the special scenario, where an email recipient intentionally wants to receive as much email as possible, such as contact email addresses used by corporations for receiving inquiries, that email account could be deliberately configured not to have a whitelist and allow email from anyone. But it would still perform the retrieval of email by polling from STS, to ensure senders cannot use fake sender addresses that use an incorrect sender domain.

## Compatibility

The contents and the structure of email can remain unchanged.
The SMTP protocol should get enhanced to support the new notification and polling mechanisms.
SMTP servers must be enhanced to support processing of notifiations, the RCA list, and the queueing of messages until they are fetched.
The IMAP protocol should get enhanced to allow the configuration of the permitted whitelist and the retrieval of the RCA.
SMTP servers could run in a backwards compatibility mode, and continue to accept non-whitelisted email as long as deemed necessary by the administrators of the RTS system.

For an RTS that supports the new protocols, it might decide to accept mail from legacy servers, but not deliver it into the recipient's mailbox, but rather use the RCA, only. If R doesn't add S to the whitelist, the prematurely accepted email could be deleted after a configurable period.

If a sending STS attempts to send the notification and detects that the RTS doesn't support the new two-way protocol yet, it may decide to deliver the email immediately using the traditional approach.

This way, it should be possible to implement servers that are backwards compatible with today's email transmission, but already prepare for a migration to a two-way authorized mail system.

## Mailing lists

When a mailing list service sends out emails to the subscribers of a mailing list, the sender email address should be set to the list address, and other means should be used to signal the author of the email.

Each subscriber would whitelist the list's sender address, and would therefore be able to receive emails from any mailing list members.

Subscribed list members are whitelisted, and incoming email from list members will be accepted for distribution.

A mailing list administrator has the responsibility to verify new list members, and remove list members that abuse their membership to send unwanted spam to the list.

## User interaction

E-Mail web interfaces and E-Mail software needs to implement management of the whitelist of acceptable senders, and integrate (on demand) display of the RCA into the user interface.