Advantages and vulnerabilities of pull-based email-delivery

Natascha Chrobok **Andrew Trotman** **Richard O'Keefe**

Department of Computer Science
University of Otago,
PO Box 56, Dunedin 9054, New Zealand
Email: nat@cs.otago.ac.nz, andrew@cs.otago.ac.nz, ok@cs.otago.ac.nz

Abstract

Over the last decade spam has become a serious problem to email-users all over the world. Most of the daily email-traffic consists of this unwanted spam. There are various methods that have been proposed to fight spam, from IP-based blocking to filtering incoming email-messages. However it seems that it is impossible to overcome this problem as the number of email-messages that are considered spam is increasing. But maybe these techniques target the problem at the wrong side: it is the email-delivery protocol itself that fosters the existence of spam. What once was created to make internet-mail communication as easy and as reliable as possible became abused by modern day spammers. This paper proposes a different approach: instead of accepting all messages unquestioned it introduces a way to empower the receiver by giving him the control to decide if he wants to receive a message or not. By extending SMTP to pull messages instead of receiving them an attempt to stem the flood of spam is made. The pull-based approach works without involvement of the end-users. However this new system does not come without a price: it opens the possibility of a distributed denial of service (DDOS)-attacks against legitimate mail-transfer agents. This vulnerability and possible ways to overcome it are also discussed in this paper.

Keywords: Spam, SMTP, Pull-based email retrieval, Denial of Service

1 Introduction

Since the early days of the internet, email has been an important part of electronic communication between people. While there are numerous ways to exchange information, email still seems to be one of the most popular ways to communicate. Used in private as well as in business environments, electronic mail became an important part of our daily life. But with all the positive aspects of communication, communication with email has its dark side: spam.

Over the last decade unsolicited bulk email, commonly known as spam, has become an increasing problem to email-users all over the world. While in the beginning it was just annoying to delete all the unwanted email-messages in the inbox, the public perception of spam changed dramatically.

But these unwanted email-messages are not only annoying end-users, they also cost tremendous amounts each year. Companies and public institutions are spending considerable effort and money to find ways to stem the further spread of spam. The yearly costs of spam are estimated to be as high as US-$50 billion (Ferris-Research 2005). However it seems that no matter how hard we try the spammers always seem to be one step ahead. There are estimations that approximately 90 percent of all email messages could be considered spam (Symantec 2009). Spam costs businesses a lot of wasted bandwith that could be used elsewhere. It can be considered wasted because it is used for receiving data that will most probably be deleted after it has been received. Also CPU time on mail-servers has to be devoted to process and filter all the incoming messages.Yet there is no guarantee that all unwanted messages will actually be filtered. Those spam-messages that get through to the end-user still need to be deleted manually which costs precious human time. However there is still the risk that regular email-messages might be mistakenly recognized as spam.

In the past few years a vast number of proposals to prevent spam have been made. However it seems that these countermeasures are not effective as spam is still with us. It is as if one of the blessings of the information-age became the ultimate curse: email-users (both corporative and private) all over the world find themselves in the grasp of spammers.

In this paper a closer look is taken at the various techniques of spam prevention. A definition of spam and its origins is given. It discusses the advantages and disadvantages of SMTP and shows how this protocol can be abused. As a huge amount of email-spam originates from illegitimate sources like the botnets, a suggestion to extend SMTP by adding a pull-based approach to make it more robust against misuse is made, and possible ways to introduce the extension are discussed. As the pull-based approach might be vulnerable to distributed denial of service (DDOS) attacks, this issue is discussed and possible solutions to overcome this vulnerability are given.

1.1 What is spam?

Unsolicited bulk email (UBE), also known as email-spam, comes in the form of email-messages for which the recipient has not granted verifiable permission to be sent and which are sent as part of a larger collection of messages (Spamhaus-Project 2009). However this definition is not precise because there also exists spam related to SMS, IP-telephony, chats, web-forums - there even exists YouTube-spam. What all these versions of spam have in common are the following characteristics:

1. Spam comes in the form of electronic messages, which are
2. sent in bulk and are

Copyright ©2010, Australian Computer Society, Inc. This paper appeared at the Australasian Information Security Conference 2010 (AISC2010), Brisbane, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 105, Colin Boyd and Willy Susilo, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

3. unsolicited.

This definition is valid for all types of spam. However in this paper spam is restricted to email-spam.

1.1.1 Types of Email-Spam

Email-spam comes in different types. According to the intention of the spammer the bulk-emails can be categorized in different ways. Over a one month period (01.08.2009 - 31.08.2009) the universities spam-filter (PureMessage by Sophos) filtered 1471 spam-messages which were dedicated for one of the author's email-addresses (see Figure 1).

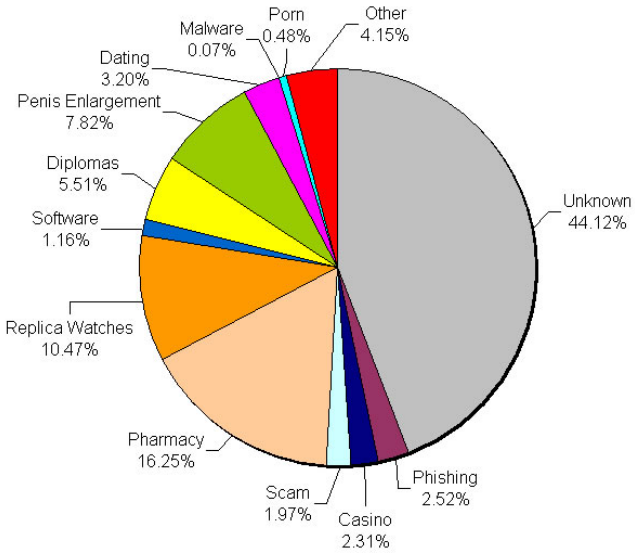

Figure 1: Spam received August 2009

These received spam-messages mostly consisted of advertisements. But there were also fraudulent messages (scam & phishing), messages containing links to adult content as well as a message containing malicious software. Most interesting was the fact that more than 44% of all received spam-messages were not intended for english-speaking recipient and without proper text-encoding (which resulted in unreadable junk-messages).

Based on the received messages, spam can be categorized as

1. advertising
2. fraudulent
3. malware-spam

The best known versions of spam are for pharmacy products or replica watches, university diplomas, online-casinos and offers for the enlargement of certain body parts. But not all advertisements need to be of commercial nature. Sometimes spam is used to propagate political or religious ideas.

The second category is that of fraudulent emails. It consists of spam messages that target naive email-users in the form of scam or phishing-messages. Unlike commercial spam the purpose of this kind of unsolicited email is criminal. Usually the goal is to get money from the recipient - either by persuading them to send money or to reveal their bank or creditcard-information. The most popular scam emails are the so-called 419-scam messages (Levy & Arce 2004) – 419 is the international prefix number of Nigeria where most of these messages originate.

The last category is malware-spam. Its main purpose is to install malicious software on the recipients computer. This malicious software could be used to gather information like email-addresses from the victims PC or it could turn the computer into a so-called zombie-PC - a remotely controlled computer which is part of a global network of hijacked computers, called the botnets. The computers in a botnet are used to send spam, host phishing sites or for cyber-warfare.

1.1.2 Distribution of Spam

The way spam is distributed has changed significantly over the last decade. While before spammers used their own mail-servers to send spam, legislative anti-spam measures such as the can-spam act 2003 and efforts from the anti-spam community have forced them to use different ways to distribute their emails. They reacted by either moving their servers into off-shore countries that have no anti-spam legislation or by abusing badly configured, third-party mail-servers. Recently the distribution of spam via the botnets, vast armies of remotely controlled zombie-PCs has dramatically increased (Herley & Florencio 2008). Estimations are that about 80% of all spam messages sent are originating from the botnets (MessageLabs/Symantec 2009). So it is essential to take the botnet as spam-generator into consideration when searching for ways to reduce the number of unsolicited email-messages.

1.1.3 Why do spammers send Spam?

A lot of spam-messages are filtered or deleted and only a very small percentage of it reaches its recipient. Nevertheless spammers tend to send millions of messages each day. The reason why they still send advertisements is that there are some people who are buying those products. It is believed that for 100 boxes of Viagra sold a spammers margin of profit could be between US-$ 1,000 and US-$ 2,000 (Spammer-X 2004). Even if it is very annoying for most of the email-users to have their inboxes flooded with unwanted email-messages, it is this tiny percentage of people who positively respond to spam to keep spammers sending. According to the results of a study carried out by researchers from University of California, Berkeley and UC, San Diego (UCSD) the revenue rate of spammers is very low compared to the number of spam sent (Kanich et al. 2008): by getting control over parts of the Storm-botnet they were able to monitor the distribution of three spam-campaigns over the time period of 26 days. During this period more than 350 million email messages containing pharmacy-related spam were distributed resulting in only 28 sales (roughly 0.00001% of all spam-messages sent). Thus the average daily revenue rate was about US-$100. However the researchers controlled only a small part of the botnet (about 1.5% of all worker-bots), so they estimate the daily revenue revenue rate for the whole botnet is about US-$7,000. Considering the large number of spam-messages sent and the small number of sales leads to the conclusion that the only way for spammers to increase their profits is to send more spam.

2 How to abuse SMTP

2.1 Overview

One of the reasons why email became so popular is the way messages are sent via the Simple Mail Transfer Protocol (Klensin 2008). The original standard for message forwarding via SMTP was created in 1982

and after nearly 30 years mail-providers still use this protocol. There have been extensions to the protocol over the years to provide new functionality but SMTP is still essentially the same protocol as it was in the beginning.

Unlike other internet protocols which are pulling information from servers SMTP works the other way round: it pushes messages from the sender to the receiver. While we know exactly which website we want to browse we usually have no idea who wants to send us mail. Therefore SMTP leaves the responsibility that a messages reaches its recipient with the sender. In the early days of internet-mail this was acceptable. There was a limited number of users and everybody could be trusted. Most of the users were members of universities or government agencies. It was unthinkable that one of the users of the email-network would abuse the system for his personal gain. So there was no use for more sophisticated security features. It was more important that a message was delivered to its recipient. The original design of the protocol lacked proper security features that would make it more resistant to misuse.

2.2 Simplicity of SMTP

The advantage of SMTP always was its simplicity. However exactly this simplicity eventually led to the problems with spam we have today. Anybody can send a message - whether it is wanted or not. The receiver then has to decide if he wants to read the message or not. This is because there is insufficient information about the content of an email available before the data is sent.

An email consists of two main parts: the envelope and the content. The envelope consists of the email-address of the originator, any number of recipients and optionally additional information for protocol extensions. The content is sent in the SMTP-DATA protocol unit and describes the actual internet message as defined by RFC 2822 (Resnick 2001). It includes the subject, the body of the message, any attachments as well as meta-information (such as sender, recipient(s), sent and received dates, or any other useful data) in the message header.

SMTP uses a small set of commands which makes it easy to implement. It is possible to send an email message without using a mail-client or a mail transfer-agent (MTA). By using telnet it is possible to connect to a mail-exchange server and to send a message by simply typing the correct SMTP-commands in the right order. If the content of the message looks legitimate, the chances are high that it will be delivered (and not filtered out). It is exactly this simplicity that makes SMTP so vulnerable and allows spammers to abuse it to distribute their messages.

2.3 Sender-Information provided by SMTP

Mail transactions in SMTP consist of three steps: the MAIL-command which specifies the sender identification, one or more RCPT-commands providing the receiver information and the DATA-command followed by the actual email-content. The information SMTP provides before the DATA-command is limited. A receiving mail transfer-agent (RMTA) only knows the following information:

- IP-address of the sending mail transfer-agent (SMTA)
- the phrase the SMTA authenticated itself with using the EHLO command (this might be a domain- or computer-name but could also be a random sequence of characters)
- the senders email-address (which does not necessarily have to be correct)
- the email-addresses of the recipient(s)

Of all this information the IP-address is the only reliable information. Of course even an IP-address can be spoofed (Savage et al. 2000), but in this case the senders IP-address is needed for protocol-communication. However all the other information provided by the SMTA can be faked.

According to RFC 5321 the SMTA has to identify itself with the EHLO-command. Usually this identification is a full qualified domain name, so the identification might look like

```
EHLO example.com
```

However it is legitimate to use other address literals if a domain-name is not available for whatever reason. Other legitimate address literals are IPv4-addresses (enclosed by brackets), IPv6-addresses, and other ways of addressing. As there are so many legitimate ways a SMTA might identify itself it is nearly impossible to determine whether or not a given address literal is valid. Most SMTP-servers therefore accept any combination of literals in the EHLO command. So for instance an identification sequence like

```
EHLO spamspamwonderfulspam
```

would be accepted by most mail-servers. Thus the identification provided in the EHLO command is of no value for the RMTA.

The senders email-address is also of no use to determine if the email is originating from a trustful source. Every valid email-address is allowed (and the address does not need to match the one stored as sender-address in the email-message). The purpose of this address is to have a return-address in case a message could not be delivered. The notification that there was a problem can be sent to this address. This cannot be used as information to find out whether the senders address is valid or not.

The next information provided is the email-address of the recipient (or the addresses if the mail is sent to several recipients). It is possible to check if a given receivers email-address exists on the receiving mail-server so it is possible to reject a message if the receiver is not valid. This procedure is usually not used for security purposes as spammers often send emails to randomly generated addresses. It would be very unwise from a IT-security stance to reject invalid recipients as spammers then could find out which email-addresses are valid by using trial-and-error systems.

So the only way to determine if the sender is trustworthy at this point is to use the IP-address. The RMTA could check in a white- or blacklist if this IP-address can be trusted or not. However there is still the chance that the SMTA is working as a relay and thus not the originator of the message. Not even the IP-address therefore could be used as a way to identify if the message is from a trusted source.

Currently the only way to determine if an email-message is spam or not is to receive the whole content of the message. The various parts of the message can then be analyzed by spam filters. Unfortunately this means that the whole SMTP-process has to be performed, i.e. the whole message has to be received. In the case of a spam-message (which would be dropped immediately if recognized as such) this is both a waste of bandwith (for the delivery-process) as well as storage (on the RMTA until the filtering has been done).

Method	Side	Effect
TCP-Blocking	Sender	Blocking the ports usually used for mail-transmitting
Limitation of Emails	Sender	Limiting the number of emails that can be sent
Micro-payment	Sender	Charging a small fee for each outgoing email
Blacklisting	Receiver	Using a list of IP-addresses which should be blocked
Whitelisting	Receiver	Using a list of IP-addresses that are allways accepted
Greylisting	Receiver	Delaying the mail-transfer by rejecting the first connection
Authen-tication	Sender/ Receiver	Incoming connections must be authenticated first before any mail-traffic happens
Challenge/ Response	Sender/ Receiver	Sender must correctly respond to a challenge sent by the receiver
Filtering	Receiver	Analyzing the messages to determine if they contain spam or not

Table 1: Methods to overcome spam

3 Review of methods to overcome spam

There are several methods to overcome unsolicited emails(Hayati & Potdar 2008). Table 1 shows an overview of the most popular methods. They are used to fight spam at different stages of the email delivery process. There are some measures that can be applied at the senders side, but the majority of the anti-spam methods are on the receiver side. This is mostly because the spammers have more control of the sender side. However this does not mean per-se that spam could not be prevented at the beginning of the delivery process. Such methods can only be applied if the ISP on the sender-side is willing to apply them.

3.1 Sender-side methods

3.1.1 TCP-blocking

One method of spam prevention is TCP-blocking. In this approach the ISP blocks TCP-port 25, which is the one used by the SMTP protocol. This makes it impossible for clients in the ISPs network to send email-messages via this port. So this simple method can prevent infected zombie-PCs from sending spam. However it makes it impossible for clients in this network to run their own valid mail-servers or to connect to other SMTP-servers (such as freemail-services like gmail). For this reason most ISPs refrain from using this method.

3.1.2 Limitation of emails

Another way to prevent spam on the sender side is by limiting the number of emails a client can send in a certain period of time. So an ISP could impose a limit of 100 outbound email-messages per day which would be more than enough for the majority of its users. If an email-client exceeds this limit, the ISP could either deny sending the message or inform the client that he or she exceeded the limit.

3.1.3 Micropayment

Micropayment is a system in which every time an email-message is sent, the sender is charged a small amount of money, for example 0.0001 Dollars. While this amount is so small that it would be negligible to regular email-users it would be very expensive for spammers sending millions of emails every day. As charges can be reckoned by ISPs this approach is possible, however it does not take into account what happens to unsuspicious users whose PCs have been hijacked.

3.2 Receiver-side methods

3.2.1 Black and Whitelisting

Another approach is IP-based blocking. When a mail-sending host connects to the RMTA the first information the receiver gets from the sender is his IP-address. Spammers might use a number of ways to hide their identity, but they have to give away the IP-address. To make bidirectional IP-based communication such as SMTP possible the receiver needs to know the IP-address at the other end.

This information is used to determine if an SMTP-session with the sender should be accepted or not. There are two ways to use this information: black- and whitelisting. Blacklisting is to determine if the IP-address of the connecting host has sent spam in the past. This is achieved by querying a list of IP-addresses. These lists could be maintained by ISPs or by anti-spam organizations that provide them to third-parties. Some blacklists like the Spamhaus block list are DNS-based. Blacklists contain IP addresses of hosts of known spammers, open relays and proxies. If the IP-address of the connecting host is on this blacklist, the connection is refused and no email-data is received.

Whitelisting is exactly the opposite of blacklisting: a list of trustworthy IP-addresses is used to determine if an incoming request is from a trustworthy source or not. It is not uncommon to use both, black- and whitelists in combination. The problem with using lists is that they tend to be large and need to be kept up-to-date.

3.2.2 Greylisting

Greylisting is based on the reliability of the SMTP-protocol: SMTP-servers will try to resend an email if an attempt to do so fails. It is assumed that the software used by spammers has a lax implementations of the standards, so they might not resend an email if the first attempt to deliver the message did not work. Greylisting also makes use of black- and whitelists

and it has proven to be a quite effective way to protect email-servers against the flood of spam. Combining greylisting with black- and whitelisting appears to be a very effective way to prevent spam. Unfortunately it can also block regular email, e.g. when the sending host is part of an email-cluster with different IP-addresses.

3.2.3 Authentication

Authentication for email-delivery is often used by ISPs and freemail-providers. Until recently it was common to have open SMTP-servers which could be abused for sending emails. This has changed with SMTP-extensions like SMTP-AUTH (Myers 1999) which require the email-user to authenticate with a username/password combination before the SMTP-server can be used. Authentication is a successful method on the sender-side to prevent spammers from using an SMTP-server.

It is very popular for spammers to forge the sender's email-address in the email envelope. There was need for a technique to prevent this forgery and to make it impossible to abuse the email-addresses of unsuspecting victims. The solution for this problem is the sender policy framework (SPF), which prevents the forgery of email-addresses. It is possible to store the SPF-data in the Domain Name Service (DNS) TXT-entries. This allows receivers to find out which hosts are allowed to send emails for a domain by making a simple DNS-query.

3.2.4 Cryptographic authentication

In cryptographic authentication, a digital signature is added to an email-message. A popular approach is the Domainkeys identified Email (Allman et al. 2007) which attempts to prevent spammers from forging source-domains. This allows domain-based black- and whitelists to be more effective.

3.2.5 Challenge/response mechanisms

Challenge/response uses a form of verification mechanism to determine if or not the sender is legitimate. Incoming messages from unverified sources are held in a queue and a challenge is sent back to the sender. This could be a simple mathematical problem (e.g. 5 + 4 = ?) or a CAPTCHA-picture. A legitimate user can respond to this challenge by sending a solution back to the receiver. While this method can be very effective against spam it might make email-communication confusing to some end-users as they don't expect to solve puzzles when sending an email.

Automated mailing-services like mailinglists, newsletters, etc. cannot respond to challenges. If both, sender and receiver use challenge/response mechanisms it could happen that challenges sent by the one result in challenges by the other and thus an endless loop of challenges is created.

3.2.6 Filtering

One of the most successful attempts to attack spam is the use of filters. There are numerous approaches to filter messages (Cormack 2006): Rule-based filters use a large set of freely configureable rules to determine if a given email contains spam. Bayesian filter systems calculate the probability of an email-message being spam while signature-based filters make use of methods such as hash-algorithms to find out if a message can be trusted or not. Modern spamfilters are highly sophisticated programs that use a combination of these three techniques and have a high rate of success at finding spam.

But spammers are always finding ways to prevent their messages from being filtered. The crux of the matter is that filtering means that we simply accept spam flooding our inboxes. Filtering might relieve the end users from huge numbers of unwanted messages but it still means that spam uses bandwith as well as storage and CPU-time at the receiving mail-servers. It is desirable to prevent spam without brute-force filtering every incoming email-message.

4 A pull-based strategy to prevent spam

The biggest problem in successful spam-prevention is the SMTP-protocol itself as it fosters its own abuse. In most cases the spam-email has been received and the only thing to do is to limit its effects to the end-user by filtering or finding out if the sender is trustworthy. And even the methods that try to solve the problem at the beginning of the email-delivery process are attempts to compensate the deficiencies in SMTP. The big question is: why are people still using a nearly thirty year old protocol when it is the source of all the trouble?

SMTP does a good job at delivering emails, even if exactly this advantage is also abused by spammers to deliver their spam. The other reason SMTP is still in use is that it is one of the most used protocols on the internet. Given that email is a vital part of daily communication both for business and private users, it is difficult to imagine a world without email. The wide use of SMTP makes it hard to be replaced with a newer, more secure protocol. Billions of internet-users expect their emails to be delivered, regardless of which system is used to deliver the message. Planning a replacement would require a world-wide agreement that SMTP has to be replaced. Even if this task were successful there is still the question of how to replace it. There is no question that there is need of a transitional period of several years in which both the old and the new protocols would co-exist. A first step could be an agreement between large freemail-providers like Yahoo!, Google, Microsoft, etc. and the major ISPs to replace SMTP. The optimistic assumption is that after planning the transitional period more and more email-service-providers would jump on the bandwagon as they won't want to be locked out from global communication via email.

But what would a replacement to SMTP look like? A new protocol should have the same features as SMTP, but without its drawbacks:

- it should be easy to use for end-users
- it should be compatible with existing internet-email standards
- it should reliably deliver legitimate emails
- it should be difficult to abuse this system

the first feature is essential: email-users should not be bothered with any changes to the delivery-protocols. Users should be able to use their email-client with all the functionality they are familiar with. They should not be forced into using new email-applications just because the new standard is not supported by their preferred software. It would be advantageous if a new standard would allow legacy clients to use it - so this would mean no change for the end-user. There should also be no change to the way an email-message is presented to the end-user. Email-address of sender and recipient, subject and body of messages should look exactly the same as they looked in the past.

The other three features are of significant importance for the new protocol: the new protocol

should not introduce new, incompatible mechanics but should work with existing standards. SMTP provides a perfect set of commands to deliver email-messages, so a new approach should be based on existing functionality and extend it instead of using different techniques.

Like SMTP it should be possible to reliably deliver email messages to the recipient. The sender of an email-message should expect that a message will be delivered to the receiver and if this is not possible he should be informed that there was a problem in sending the message.

But unlike SMTP the new protocol should only deliver email-messages originating from a trusted source. Spammers should not be able to use the new protocol the same way they misuse SMTP.

4.1 Pull instead of Push

Internet-email, unlike many other internet-based protocols, is a push-based protocol. This means that communication is initiated by the sender as the receiver does neither know about a message he will receive nor when this message will be sent. But it means that the receiver has to accept all incoming messages, regardless of its content. During the whole delivery-process the control lies in the hands of the sender. The receiver has little influence in this email-delivery process - as long as the sender is a trusted source this procedure is acceptable. But it also makes the receiver vulnerable should the sender abuse this system. There are few methods that give the receiver control over the delivery-process (such as black- or whitelisting).

Pull-based protocols work the other way round. The receiver initiates the communication by requesting information. Thus he has more control over which information is received whereas the sender is only the provider of this information. There are numerous pull-based protocols; with HTTP the most obvious. Using a pull-based approach for internet-email empowers the receiver and gives him control over the delivery-process as he can decide when and what he wants to receive.

Pull-based email was first introduced with Internet Mail 2000 (Bernstein 2000). Instead of forwarding every email-message automatically a notification that there is email available is sent to the recipient. The receiver then decides if he wants to receive messages from this sender or not. In a positive case the receiving MTA pulls the email from the sender. If the recipient does not want to receive mail from the sender, the notification is simply ignored and not responded to. The difference between this approach and SMTP is that during the whole process the email-data is stored at the senders side. This is important as it becomes possible for the recipient to decide which email-messages he wants to receive (and which not) instead of blindly accepting and filtering every incoming message. Although Internet Mail 2000 is simply a conceptual idea there have been several attempts to implement it. The two most notable pull-based email-services are DMTP and Stubmail.

DMTP (Duan et al. 2007) makes use of a combination of classical SMTP functionality, black-/whitelisting and a pull-based approach. By classifying senders into the categories well-known spammers, regular contacts and unclassified senders it allows the receiver to process messages in different ways depending on the sender. While messages from well-known spammers are automatically rejected those from regular contacts (stored in a list on the receivers MTA) are received using the standard SMTP-push mechanism. Messages from unclassified senders (i.e. neither well-known spammers nor in the regular contacts list) are not received, instead the sending MTA (SMTA) is notified to use the DMTP-protocol. This means the SMTA stores the message and sends a message-key and the subject of the message to the RMTA. The RMTA then generates a email-message, containing the information given by the SMTA and sends it to the end-user. Receiving this notification the end-user has to decide if he wants to receive the message or not. In the case he wants the message he responds to the RMTA. The RMTA then retrieves the message from the SMTA using the message-key and adds the SMTA to its whitelist. Future messages from this SMTA will be automatically accepted. Considering that it is unlikely that all possible spammers are stored in the blacklist and not all legitimate senders are in the whitelist means that many messages will be from a unclassified source. The result is that the end-users mailbox is flooded with email-notifications. There is a risk that the end-user accidently accepts spam-messages (resulting in whitelisting a spam-source) or rejects legitimate mails. Also the sheer mass of notifications could be perceived as annoying as spam.

Stubmail (Wong 2006) also uses a pull-based email-retrieval approach by combining classical SMTP-based internet-mail with HTTP. It checks if the receiver supports Stubmail or not and then decides how to deliver the message. If the recipient supports the new protocol extension, a key is created and a notification (the so-called stub) is sent to the receiver. Should the receiver of the message decide to read the email-message he has to pull it from the senders server. To find the address of the server on which the downloadable messages are stored the receiver must make a special DNS request. Once these address is known the message can be retrieved by an HTTP-post request. Like DMTP the number of notification could by irritating to the end-user. By using HTTP to retrieve a message it could be possible to download malicious software to the receivers PC.

Despite there being working reference-implementations Internet Mail 2000 is not universally used as a replacement to SMTP. It seems as though the proposed approaches are not perceived as replacements for the classical internet-mail system. They require too much interaction by the end-user, making it awkward to receive messages without minimizing the chance to receive spam.

4.2 General Delivery: using SMTP for Internet Mail 2000

In our new approach to Internet Mail 2000 we introduce a system which works like the snailmail-approach of *poste restante* (or general delivery). The post-office notifies the receiver that there is mail waiting at the post-office. The notification includes an identification with which the receiver can retrieve the mail from his post-office. If the receiver doesn't gather his mail within a certain time-frame the mail is returned to the sender as undeliverable.

The new functionality can be implemented as an extension to SMTP which means that existing SMTP-services could be easily upgraded to use this approach. Email-communication usually involves at least four agents: the sender's mail transfer client (Outlook, Webmail, etc.), a SMTA, a RMTA and the receiver's mail transfer client (MTC). As the communication between the RMTA and the receiver's MTC uses POP or IMAP, only the first three agents use SMTP. Unlike other pull-based approaches the only SMTA and RMTA use pull-mechanics in their communication. The decision if a message should be retrieved or not is made by the RMTA (i.e. an SMTP-server). No interaction by the end-user is needed to make general delivery work. This means that the end-user can use

the email-service as before by using their preferred client-software.

The communication between SMTA and RMTA is split into two parts. In the first part the SMTA connects to the RMTA and notifies it that an email-message is available and sends a unique identifier for this message. After this notification the connection is closed. It is then up to the receiver to decide if the message should be retrieved or not. The second part happens in the case that the receiving host decides to retrieve the message. It connects to the sender and asks for the message by handing over the unique identifier. The sender then forwards the requested message.

The general delivery extension to SMTP introduces two new commands: GDEL and RETR. One is used for delivering the notification, the other one is used for retrieving the email. As usual these extensions are mentioned in a email-servers response to the EHLO command.

The time-frame of how long a message should be stored at the sending email-host is reasonable. It should be taken into consideration that the receiving host might not retrieve a message immediately. So a time-frame between 24 hours to 48 hours might be appropriate. If a message has not been retrieved after this predefined amount of time the SMTA should send a notification to the sender of the email that the delivery of the message was not successful. Likewise the time that passes between the notification and the retrieval can be determined by the RMTA.

It might occur that a RMTA gets a notification for an email that cannot be retrieved. The reason for that could be a temporarily unreachable sending host or a corrupted unique identifier for the message. In such a case the RMTA should try to retrieve the message a number of times (perhaps 3 to 5 times) over the next 48 hours. If retrieving the message continues to be unsuccessful the RMTA discards the notification and stops retrieving it. If the message to be retrieved was legitimate (and the non-delivery was just because of technical difficulties) there are mechanics to notify the sender that the delivery of the message was not successful, which works quite the same way as the classical SMTP.

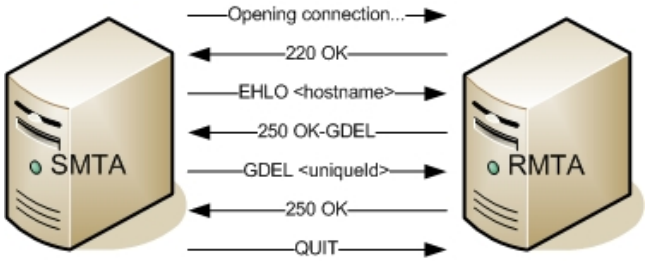

Figure 2: GDEL-command

4.2.1 GDEL (General Delivery)

The GDEL command is used by the SMTA to notify the RMTA that there is a message available. Together with the GDEL command a unique identifier for the email-message to be delivered is sent. It is up to the SMTA how to generate this unique identifier - so it could possibly use a hash-value of the email-message. It is possible to send multiple GDEL commands in the case that several messages for the same receiver-host/domain are available. In any case a unique message identifier has to be generated for every message, regardless of the fact that two or more recipients might be of the same receiver-domain. This is necessary because there is a chance that one message is retrieved while the other one is not.

When used in the response to an EHLO-command this command informs a SMTA that the receiving host is able to use the general delivery extension.

Syntax: GDEL uniqueId

Possible reply codes:

250 Requested mail action okay, completed
500 Syntax error, command unrecognized
501 Syntax error in parameters and arguments
502 Command not implemented

Figure 2 describes the usual sequence of commands of a successful notification using the GDEL-extension:

1. The SMTA opens a connection to the RMTA
2. The RMTA returns 220 OK
3. The SMTA sends the EHLO-command
4. The RMTA returns with 250 OK and a list of possible extensions supported, of which one is GDEL
5. The SMTA generates a unique identifier for the email-message and sends it using the GDEL-command
6. The RMTA stores this unique id and returns a 250 OK
7. The SMTA closes the connection by sending QUIT

It should be noted that it is possible to send several notifications in a sequence to the receiving host. After the last 250 OK the SMTA could send another notification using the GDEL-command.

4.2.2 RETR (Retrieve)

The RETR command is used by the RMTA to retrieve a message from the SMTA. The unique identifier of the email-message is passed as an argument. If the unique identifier is valid (ie. the email exists on this server) and the email is destined for the connecting host, the SMTA changes into sending mode and starts sending email using standard SMTP. It is possible to retrieve several emails in this way after one another. In the case the RMTA sends an invalid message-id, the SMTA should respond with a 550-error message. If an email-message is not destined for the connected RMTA, the SMTA also responds with a 550-error message, even if the provided unique identifier is valid.

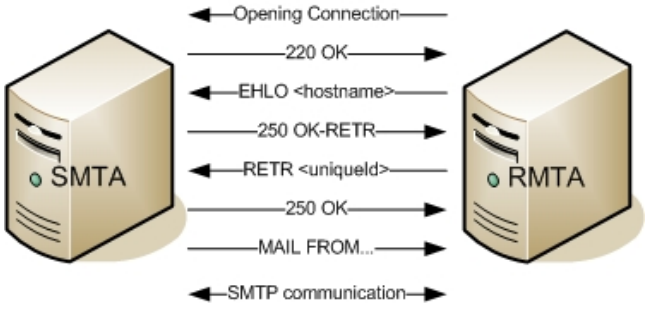

Figure 3: RETR-command

The communication for the retrieval is initiated by the receiver. When the RETR-command is sent

sender and receiver switch roles and the sender starts the SMTP-message-transfer sequence (MAIL FROM - RCPT TO - DATA). After sending the message the roles are switched again to allow the sender to send another RETR-command or to quit the connection. The original SMTP-standard had a similar command called TURN which allowed a sender to become the receiver and vice-versa (Postel 1982). However this functionality eventually became deprecated as it was possible for an unauthenticated client to retrieve messages.

When used in response to an EHLO this command informs the RMTA that the sending host is able to use the general delivery extension.

Syntax: RETR uniqueId

Possible reply codes:

250 Requested mail action okay, completed
500 Syntax error, command unrecognized
501 Syntax error in parameters and arguments
502 Command not implemented
550 Requested action not taken (e.g. given message-Id not available)

The second part of the protocol using the RETR-extension is shown on figure 3:

1. The RMTA opens a connection to the SMTA
2. The SMTA returns with 220 OK
3. The RMTA sends the EHLO-command
4. The SMTA returns with reply-code 250 OK and also sends a list of the supported extensions (e.g. RETR)
5. The RMTA uses the stored unique id of the email-message it wants to retrieve and sends it using the RETR-command
6. The SMTA compares the requested unique message-id to a list of stored messages, if the message is available it sends 250 OK
7. Both MTA change into SMTP-mode and the SMTA starts forwarding the email-message using the MAIL FROM-command
8. subsequent communication is classical SMTP

As soon as a message is retrieved by a client, there is no need for the SMTA to store it any longer. All subsequent requests for an already retrieved message should be denied with return-code 550 Requested action not taken. It is important for SMTP-clusters where a retrievable message could be requested from several servers is made invalid on all servers belonging to the same cluster. It should be not possible to retrieve a message from a server which has already been retrieved on a server in this cluster.

4.3 Combination with other techniques for spam-prevention

In order to prevent spam effectively it is important to combine several techniques. The general delivery extension to SMTP does not prevent the use of other methods to make the mailing process more secure. It can be combined with other techniques like white- and blacklists to make it easier to determine the senders credibility. By using IP-based lists at the initiation of the communication process it is possible to prevent connections from illegitimate hosts while allowing host that are on the white-list to connect without problem. This could be used in a way to selectively decide if the general-delivery extension should be used for a connecting host or not. Using black- and white-lists that way is a convenient way for legitimate email-users as the flow of their messages is not disturbed while it is more difficult for spammers to get their emails through.

On the sender side, ISPs could use TCP-blocking to prevent outgoing email-connections to hosts outside of their own network as well as incoming retrieval requests. Many ISPs don't allow the operation of network-based servers for consumer-connections. Power-users and corporate customers could have special contracts with their ISPs that allow them to operate those services. This would make it very hard for infected zombie-PCs to provide SMTP-based services. However this method needs the cooperation of ISPs and it is questionable whether a worldwide agreement could ever be reached. Finally the use of general delivery extension does not exclude the use of filter-software on the receivers side. Although it makes it harder for spammers to get their mail through it does not prevent spam. And there is also the chance that legitimate and therefore trusted hosts might have been compromised and send spam. So the use of spam-filtering complements the mix of effective tools to prevent spam in this scenario.

4.3.1 Advantages

For spammers to be successful it is essential to send as many messages as possible in a short period of time. Using general delivery forces them to store all the messages that they want to send on their server until they are retrieved. Regardless of whether the messages are sent from an email-server in an offshore country or by a zombie-PC in a botnet this means that the SMTP-service must be provided for an undefined timeframe as the spammer does not know when the receiver will try to retrieve the message (if he ever does so). Especially hijacked PCs in a botnet need to stay longer online as they must provide the SMTP-service for incoming retrieval requests. Botnets tend to send their spam in bursts (Xie et al. 2008) – after a period of inactivity the bots are activated and start sending spam for a certain amount of time (usually a couple of hours) followed by another period of inactivity. This time of inactivity is used for botnet-maintainance during which bots might get new instructions, new lists of email-addresses and material for new spam-campaigns. Using the pull-based approach it becomes more awkward for spammers to use bots for propagating spam as they cannot use the hit-and-run tactic anymore. By providing the general delivery service the bots must stay active all the time. This could also become a problem for a consumer-based internet-connection when a lot of retrieval requests are incoming. So one could say that general delivery does not prevent spam, but it makes it more difficult for spammers to get their messages through. The separation of the delivery process into two parts gives the RMTA more time to decide if a message should be retrieved or not. During the time between notification and retrieval the RMTA could run processes to find out if the sender is a trustworthy source. Most notably general delivery is an extension to the standard SMTP-procedure, so both the new mail-service as well as the classical service can be serviced by the same mailing host. This is an advantage during the transitional period as only one service needs to be maintained instead of two parallel running services. After the transitional period

the classical SMTP-functionality could simple be deactivated. General delivery works on protocol-level between email-servers, so no interaction by the end-user is needed. This makes it very convenient as there is no change for the user.

4.3.2 Disadvantages

The new mechanism comes with some disadvantages. The most obvious is the protocol overhead. The communication between SMTA and RMTA produces more traffic than standard SMTP communication. There is the notification and a retrieval. In the case of a successful message-delivery the amount of data transfered between sender and receiver is larger than with the classical protocol. However it should be taken into consideration that not every email-message will be retrieved as the RMTA decides that it comes from an untrustworthy source. Considering the vast amount of spam that will not be transfered we believe it is more than a cheap payoff and therefore worth the additional traffic for legitimate email-communication.

Spammers using a botnet could provide SMTP services on hijacked PCs. However the time beween notification and retrieval could be used by the RMTA to determine whether that the sending email-host is a legitimate email-server (i.e. by querying a blacklist, etc.) and thus decide not to retrieve the email.

Another argument against the general delivery extension is that it could be used to make DDOS attacks. This vulnerability will be discussed in detail in the next section.

4.4 Vulnerabilities

Using the pull-based approach for internet-mail has many advantages. The most important of which is that the responsibility for email-storage is moved from the receiver to the sender and that the receiver can decide if he wants to retrieve the messages. However these advantages do not come without a problem. Unlike ordinary SMTP-based services the vulnerability does not lie on the receivers side but on the senders. The pull-based approach makes it necessary for the SMTA to act as a client (when sending notifications) and a server (when providing services for incoming retrieval-requests). During the notification the SMTA has control over the process as it initiates it. On the other side the SMTA has no control over who connects during the retrieval-process. As long as a legitimate client connects everything is fine. However there is the possibility of incoming connections that might not be according to the protocol. The last problem might be a misconfigured RMTA that is repeatedly trying to retrieve a non-existing or already retrieved message.

As retrievable messages are identified by a unique id it is possible that third parties try to illegitimately attain messages by simply guessing the unique id. Particularly implementations with open sources could make this process possible as attackers could write scripts that send retrieve-requests with randomly generated id-keys that are according the key-generation-algorithm. Though the chance to retrieve a particular message is not high, there is still the possibility of generating a valid key. As a certain message can be only retrieved by the RMTA for which it is destined, an attacker needs to pretend to be the correct receiving host. This makes it extremely hard to randomly retrieve messages, but if the attacker has knowledge of emails on the SMTA that are destined for a specific receiver domain, it could be possible to retrieve messages by a brute force attack. Even if the chance of actually retrieving messages is not high it could easily lead to a performance problem as the SMTA has to process a lot of unnecessary requests. Especially a combined brute-force attack of retrieval requests could lead to a denial of service as the SMTA is unable to process all the requests at once. For this reason it is suggested that further communication is delayed for a reasonable time before another attempt to retrieve a message is possible. There remains the possibility of adding hosts that continuously try to retrieve non-existant messages with wrong message-ids to a blacklist.

The greatest threat to the pull-based approach is that it could be used to intentionally attack a mail-server with a distributed denial of service attack. It is possible to use the message-notification mechanism to force a large number of RMTA to make retrieval-requests even if there is no message actually to be retrieved. So an attacker could send millions of notifications to different RMTA, notifying them that a message is available on the email-server of domain victim.com.

A botnet of tens of thousands of zombie-PCs might generate a tremendous amount of email-notifications. The RMTA will try to contact the mailing-host on which the message is apparently stored. Though there is no guarantee when (and if) RMTA will try to retrieve a message it is obvious that a huge number of requests could easily bring down an SMTA.

One effective way to reduce the vulnerability to DDOS-attacks is to make sure that the notifications are sent from a trusted source i.e. the notification comes from the same address as the message. This means the receiver of a notification has the responsibility to determine whether a notification is from a legitimate origin. So when there is an incoming notification, the receiver should query the IP-address of the connecting host and find out if it belongs to the number of hosts that are allowed to send emails for their domain.

The best way to find more information about the sender of a message is use the DNS. Many domains have an MX-record used to determine which hosts are responsible for mail-exchange. Though usually the MX-records are used to determine the mail-hosts for incoming email-traffic on a domain, the change to a pull-based approach brings to the mail-protocol a means by which they could be used for both in- and outgoing traffic. Using the MX-record would be a misuse of the DNS functionality, but it could be argued that the goal of the pull-based approach is to replace the traditional way to exchange email-messages and so it justifies its use.

But there is another way to determine the identity of the sender - the Sender Policy Framework (Wong & Schlitt 2006). The intention of the Sender Policy Framework (SPF) is to prevent the forgery of email-address senders by explicitly authorizing the hosts that are used for mail-transfer. Using SPF, receiving hosts can query to determine whether a connecting host is authorized to send emails (or notifications in the case of the pull-based approach). SPF makes use of the DNS-protocol TXT entry which is used to store arbitrary text-based attributes. It is possible to define which hosts are allowed to send emails on behalf of a domain. Hosts do not necessarily need to be in the same domain, SPF allows authorizing mail-hosts belonging to other domains. A possible SPF entry for using the pull-based mail-service could look like this:

```
example.com.  TXT "v=spf1 mx
a:pullmail.example.com -all"
```

For the domain example.com all outgoing MX-servers are authorized to send (and provide) emails as well as the mailhost pullmail.example.com. All other

hosts are not allowed to send or provide emails on behalf of this domain. When getting notification from an SMTA, all the RMTA has to do is to query the SPF-record for the domain the notification is from and compare it to the IP-address of the connecting host. If the host is in the list of authorized mailservers, the RMTA can proceed to retrieve the email. In any other cases the notification can simply be rejected. Using this technique is very effective against connections from a botnet because it is unlikely that a zombie-PC has a valid SPF-record. And even if there are SPF-entries for botnet-hosts, the SPF-query can still be combined with a blacklist-query and spam-filtering to make it more effective.

5 Conclusions

This paper discusses the current protocol for internet-email, SMTP and why its architecture (which is focused on reliability and simplicity) fosters the spread of unsolicited email. The various methods and techniques to recognize and prevent spam, both on the sender and the receiver side have been presented.

Especially in a time were huge amounts of spam-messages originate from hijacked botnet-PCs it is important to find new ways to make the distribution of unwanted messages harder. Therefore a pull-based approach to retrieve emails, which is in contrast to the classical push-based approach is suggested.

One advantage of the pull based approach is that the responsibility to store the email-messages is transferred from the receiver to the sender. As the receiver just gets a notification, that there is a message available to be retrieved at the sender's server, it is easy for him to decide if he trusts the sender and gets the email or to just ignore the notification. Therefore bandwidth can be saved as not every message has to be received. The pull-based approach, called general delivery, makes its use as well as its introduction very easy. By just adding two new commands to the set of existing SMTP-commands the new email-pull functionality is provided. As the pull-based approach needs no user-interaction, it can be introduced without end-users interaction.

However the pull-based approach comes not without disadvantages. Most notably is its vulnerability to distributed denial of service attacks — when an attacker sends notifications to various mailservers that messages can be retrieved at a certain host. This vulnerability can only be reduced on the receivers side by making sure that the notification has been made by the correct host and not by a third party pretending to be the sender. To make this possible, existing techniques like the DNS-entries of the Sender Policy Framework could be used. The new approach is a way to stem the flood of spam in emails. However it is clear that an effective solution to spam must be a cocktail of various anti-spam measures.

References

Allman, E., Callas, J., Delaney, M., Libbey, M., Fenton, J. & Thomas, M. (2007), 'RFC 4870 Domainkeys Identified Mail (DKIM) Signatures', http://www.ietf.org/rfc/rfc4870.txt.

Bernstein, D. J. (2000), 'Internet mail 2000', http://cr.yp.to/im2000.html.

Cormack, G. V. (2006), 'Email spam filtering: A systematic review', *Foundations and Trends in Information Retrieval* **1**(4).

Duan, Z., Dong, Y. & Gopalan, K. (2007), 'DMTP: Controlling spam through message delivery differentiation', *Computer Networks* **51**(10), 2616–2630.

Ferris-Research (2005), 'The global economic impact of spam, 2005', http://www.ferris.com/2005/02/24/the-global-economic-impact-of-spam-2005/.

Hayati, P. & Potdar, V. (2008), Evaluation of spam detection and prevention frameworks for email and image spam: a state of art, *in* 'Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services', ACM New York, NY, USA, pp. 520–527.

Herley, C. & Florencio, D. (2008), 'Nobody sells gold for the price of silver: Dishonesty, uncertainty and the underground economy', Microsoft Research, http://research.microsoft.com/pubs/80034/nobodysellsgoldforthepriceofsilver.pdf.

Kanich, C., Kreibich, C., Levchenko, K., Enright, B., Voelker, G., Paxson, V. & Savage, S. (2008), Spamalytics: An empirical analysis of spam marketing conversion, *in* 'Proceedings of the 15th ACM conference on Computer and communications security', ACM New York, NY, USA, pp. 3–14.

Klensin, J. (2008), 'RFC 5321 Simple Mail Transfer Protocol', http://tools.ietf.org/html/rfc5321.

Levy, E. & Arce, I. (2004), 'Criminals become tech savvy', *IEEE Security & Privacy Magazine* **2**(2), 65–68.

MessageLabs/Symantec (2009), 'Messagelabs intelligence: Q2/june 2009', http://www.messagelabs.com/mlireport/MLIReport_2009.07_July_FINAL.pdf.

Myers, J. (1999), 'RFC 2554 SMTP Service Extension for Authentication', http://www.ietf.org/rfc/rfc2554.txt.

Postel, J. (1982), 'RFC 821 Simple Mail Transfer Protocol', /urlwww.ietf.org/rfc/rfc821.txt.

Resnick, P. (2001), 'RFC 2822 Internet Message Format', http://tools.ietf.org/html/rfc2822.

Savage, S., Wetherall, D., Karlin, A. & Anderson, T. (2000), 'Practical network support for IP traceback', *ACM SIGCOMM Computer Communication Review* **30**(4), 295–306.

Spamhaus-Project (2009), 'The definition of spam', http://www.spamhaus.org/definition.html.

Spammer-X (2004), *Inside the Spam Cartel - Trade secrets from the dark side*, Syngress Publishing.

Symantec (2009), 'State of spam: A monthly report', http://go.symantec.com/spam_report.

Wong, M. & Schlitt, W. (2006), 'RFC 4408 Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1', http://tools.ietf.org/html/rfc4408.

Wong, M. W. (2006), 'Stubmail', http://www.stubmail.com.

Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G. & Osipkov, I. (2008), 'Spamming botnets: Signatures and characteristics', *ACM SIGCOMM Computer Communication Review* **38**(4), 171–182.