Restraining transmission of unsolicited bulk e-mail

W.W. de Vries
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente
w.w.devries@student.utwente.nl

ABSTRACT
Filtering large amounts of unsolicited bulk e-mail, also known as spam, is expensive. Either because of the time spent on manual deletion or complex analysis by algorithms putting heavy load on e-mail servers. Due to exponential growth of the volume of spam campaigns, Internet Service Providers (ISP’s) are increasingly forced to use rigorous rejection policies to prevent their filter servers from being overloaded. Concurrently, legitimate e-mail servers end up in blacklists due to compromised end-users. This endangers the reliability of e-mail communication and the process of automatically filtering spam.

Transmitting spam is currently easy, cheap and mostly anonymous. Restraining spam transmissions at the source network could put a halt to this; furthermore, it is less complicated and cheaper than filtering at the destination network, because of lower volumes and easier detection. This paper proposes a novel architecture designed to restrain transmissions at the source network. Experimentation suggests 75% to 100% of all spam campaigns could be restrained relatively easy and cheap if ISP’s and hosting providers would enforce simple restrictions and checks on the outgoing e-mail of their end-users.

Categories and Subject Descriptors
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—Information filtering; H.4.3 [Information Systems Applications]: Communications Applications—Electronic mail; K.6.5 [Management of Computing and Information Systems]: Security and Protection

General Terms
Security, Experimentation

Keywords
Unsolicited bulk e-mail, Spam, Classification, Filtering

1. INTRODUCTION
Unsolicited bulk e-mail (UBE), frequently referred to as spam, comprises approximately 90 to 95 percent of all e-mail traffic [20, 21]. This only includes e-mail that is both unsolicited and bulk. For example, job enquiries are unsolicited and subscriber newsletters are bulk; however, neither of them is both and thus not UBE.

Estimates of the cost of spam vary greatly from $10 billion to $87 billion annually [19]. These costs reflect lost productivity, implementation of automatic filtering techniques, and losses due to not receiving important e-mails because they were falsely classified as spam. In contrast, one single person who transmits spam campaigns can earn as much as $12 million annually [19]. Sending the campaigns costs virtually nothing; the transmission of one spam e-mail is estimated to be about 10,000 times cheaper than receiving it [10].

Consequently, the amount of spam e-mails has become astronomical in comparison to legitimate e-mail, therefore manual filtering is not an option anymore. Recipients have to resort to automatic filtering techniques, such as SpamAssassin, or client-side filters like the built-in Junk Mail Filter of Mozilla’s e-mail client Thunderbird.

The process of automatically analyzing e-mail for spam works well overall, but it has disadvantages. Most importantly, the complex algorithms to analyze the content are very resource-intensive [11]. With the vast amount of spam that is being sent today, it becomes very expensive to supply the necessary hardware to do those complex analyzes. This provokes Internet Service Providers (ISP’s) to be more aggressive about filtering by rejecting e-mails upfront if they do not comply to very rigorous requirements [13]. They do so in order to prevent overloading the servers that perform filtering through more reliable techniques like SpamAssassin. Simultaneously, legitimate e-mail servers end up in blacklists due to compromised end-users. This endangers the reliability of e-mail communication and the process of automatically filtering spam.

1.1 Problem description
Scientists conduct much research on finding methods to make filters such as SpamAssassin even more reliable and less resource-intensive. However, we recognize that exponential growth of the volume of spam campaigns requires us to take a step back and look at the greater problem: sending spam is easy, cheap and mostly anonymous. Restraining transmissions at the source network could be a crucial contribution to reduce the spam problem, because of the two following reasons:

I. Much legitimate e-mail is lost due to increasingly aggressive filtering methods by ISP’s. If the vol-
ume of spam campaigns was significantly lower, ISP’s would be less inclined to use rigorous rejection policies that are very prone to filtering out legitimate e-mail (called false positives). Reducing the volume of incoming e-mail means more e-mail could be rejected or allowed based on much more reliable decisions made by complex filters such as SpamAssassin.

II. During the transmission of e-mail, much important information is lost. For example, a recipient does not know how many copies of a specific e-mail have been sent, although this information could be useful to determine whether it is spam or not. In contrast, this information would be available to filtering technology at the source network, making it possible to prevent transmission of the spam e-mails in the first place.

Recently several governments created legislation that classifies transmission of spam campaigns as illegal. We applaud this, but we also believe that because of the nature of the internet we simultaneously need to look for a technical approach to enforce this legislation. This leads us to the following research question:

- What architecture could be implemented to significantly restrain the transmission of spam?

This raises the following additional questions:

- What methods do spammers use and what are the sources of spam transmissions?
- Which parties have the skills and knowledge to implement solutions?
- Which checks can be used to detect and restrain spam transmissions?

The novel idea of this paper is to focus on the sender’s side of spam campaigns instead of the recipient of spam. This establishes a unique view on the spam problem: we believe filter technology at the source network could restrain spam transmissions with much lower resources than filters at the recipients could. Consequently, this enables ISP’s to filter more e-mail based on reliable algorithms instead of rigorous rejection policies; significantly improving the quality of e-mail communication.

1.2 Approach and organization

In section 2 we begin with a literature study to provide for an overview of current spam methods and sources. Then we will discuss related work. Based on the findings in section 3; we select suitable types of spam and designate parties that have the skills and knowledge to implement technology to restrain transmissions of spam. We also develop a set of measurable requirements to which we can compare our proposed new architecture. In section 4 we will review existing solutions, used in practice, and rationalize why they do not conform to our requirements or are only a single element of an effective solution. We propose a new architecture in section 5, meeting our requirements, and show how involved parties could implement it. We will provide pseudo-code to support the development of a working implementation. In section 6 we will perform experiments within the infrastructure of a large Dutch web hosting provider to analyze how effective the checks of our proposed architecture are. Section 7 and 8 provide for conclusions and suggestions for further research.

2. BACKGROUND AND RELATED WORK

This section contains a literature study providing for an overview of the current state of spam and related research. We will identify the main techniques used by spammers and the main sources of spam. Next, we will discuss previous work related to our research. This literature study assists our research in determining a scope and establish a foundation to build upon, to be able to develop an effective architecture incorporating a set of checks to restrain transmissions of spam.

2.1 Background

2.1.1 Methods to send spam campaigns

Currently, four main approaches exist to easily send large amounts of spam at low cost while maintaining most or all anonymity.

Direct spammers. The easiest way to send spam is to operate and maintain your own e-mail servers. Spammers optimize their e-mail servers for the sole purpose of sending large quantities of spam. They are connected to the internet through the few ISP’s who condone this type of activity. More aggressive direct spammers use very advanced techniques to switch IP addresses on a regular basis, for example by temporarily announcing an IP address space which they do not own (through BGP hijacking [2]). Key characteristic of direct spammers is that they do not try to hide their activities and identity; they send large amounts of spam from a limited amount of IP addresses. Some of those people are publicly listed in the Register of Known Spam Operations (ROKSO) [22], so bona fide ISP’s can recognize and reject them, should they request connectivity.

Open relays. Considering how an e-mail server operates, a configuration error in the area of client authentication can lead to a so-called “open relay”. This means that anyone on the internet can use these misconfigured e-mail servers to send e-mail. This allows spammers to send their spam almost anonymously, only the operator of the misconfigured e-mail server could find out about the identity of the spammer. For example, Sendmail version 5 was an open relay by default [24]. Many unexperienced administrators make this configuration mistake, so spammers actively scan the internet for open relays. E-mail server software is fortunately increasingly configured to be secure by default, so this issue is becoming less severe.

Botnets. The modern approach to sending spam is to use botnets. A botnet consists of many bots [4], connected to a central control server maintained by a spammer. Those bots are typically Windows computers infected by a virus and owned by unaware end-users. From the control server, the spammer commands the bots to transmit his spam campaigns. This method is rapidly advancing; it provides for a cheap, distributed and completely anonymous way to transmit spam. An example of a spam tool leveraging this technique is the Rustock Rootkit and Spam Bot [3].

PHP/Perl. The second modern approach to transmitting spam is using compromised hosting accounts. Computers of end-users are first compromised through a virus infection, for example the Gumblar virus [12]. This virus exists in a modified form to steal FTP passwords and then uploads a PHP or Perl script to hosting accounts. This script contains instructions to transmit spam, and is typically activated through an HTTP call. To prevent detection and investigation after the abuse, the virus also automatically removes the script after activation. This spam method is rapidly progressing and adapting, causing severe problems for hosting providers: their servers become
listed by blacklists and are therefore unable to deliver e-mail of their customers.

2.1.2 Main sources of spam
Besides the e-mail servers of direct spammers, there are two other main sources of spam. Both sources are created by compromising and abusing the computers of others, so the actual spammers can stay anonymous and can circumvent any blacklists. Furthermore, research shows that compromised hosts tend to infect other hosts within the same network, forming clusters of so-called bad neighborhoods [17, 18]. Compromised computers can be divided into two groups:

Computers of compromised end-users. Many ISP’s allow their customers to run an e-mail server by themselves. Spammers abuse this possibility, by infecting end-users with viruses or trojans and then transforming their computers into e-mail servers that they can control from a remote location: they become part of a botnet. Since many end-users are unaware of this problem, botnets are still a very large problem. Botnets consists of hundreds of thousands of bots [4]. A large anti-spam network estimated that in June 2006 about 80 percent of all spam was sent through those botnets [7].

Hosting servers. As explained in section 2.1.1, a new botnet-like way to send spam is using compromised hosting accounts. For spammers this technique has many advantages over the conventional botnets: high-speed connections, typically unlisted IP addresses, and high availability.

2.2 Related work

2.2.1 High- and low-volume spammers
Pathak et al. conducted research on spammers’ behavior by setting up an open relay and generating statistics on the spam they collected in a period of three months [15]. During their research they observed the prevalence of two sets of spamming hosts: high-volume spammers (HVS) and low-volume spammers (LVS). The LVS is a set containing a high number of hosts, each sending a low volume of spam. In contrast, the HVS is a set containing a low number of hosts, each sending a high volume of spam. They describe the LVS as highly coordinated spamming hosts and HVS as brute force spamming hosts. Essentially, this is the distinction between the direct spammers and the botnets that we discussed in the previous paragraph.

The LVS keep a low profile by making sure that each bot sends just a few spam e-mails to each specific e-mail server, thereby avoiding discovery of their malicious activities and ultimately being listed on a blacklist. The HVS do not keep a low profile and thus get listed quickly on blacklists. Since blacklisting clearly does not work against LVS, all spam from LVS has to be filtered by solutions like SpamAssassin, causing high costs for the recipient. Moreover, LVS are responsible for 80 percent of all spam [7].

This means restraining the transmissions from LVS would be most beneficial to the internet community. It saves costs and is most effective in reducing the volume of spam, so we will focus on LVS during our research.

2.2.2 Detecting spam behavior through Netflows
Vliek researched if it is possible to detect spamming hosts via Netflow data gathered at internet routers with a low false positive rate [23]. Results are very positive: 99 out of 100 most suspicious IP’s could repeatedly be positively validated. Unfortunately his research is focused on detecting hosts that originate from external networks, to detect incoming spam. We want to detect and restrain outgoing spam. His research shows that patterns exist in Netflow data of spamming hosts, so this technique might also be interesting to detect hosts in an internal network trying to send out spam.

During our research we only have access to the network of a hosting provider, for them it is only interesting to know which specific account on a specific server causes spam. Netflow data could only tell us which server is sending spam. We are convinced Netflows could be used effectively by ISP’s to detect spamming computers of end-users, but we want to research if it is possible to implement a technique which is effective for both hosting providers and ISP’s. Moreover, Netflow has certain requirements for a network to be able to use it; many networks do not comply to those requirements.

2.2.3 Detecting spam by core attributes
In the field of spam filter research on the recipient’s side, the idea of only using the core attributes of an e-mail to filter is emerging [17]. Core attributes are described as elements of an e-mail which can not be easily modified or varied, such as source IP address or a URL the spammer wants one to visit. The theory is that all other elements can be varied indefinitely (consider the many variations to write Viagra) and thus makes content analysis complex to perform.

The concept of this idea is to choose elements of an e-mail that allow development of decisive criteria, while ensuring the criteria are light on server resources. For example, the use of URL’s registered by spammers in an e-mail tell an e-mail is spam. This check can be performed with a DNS lookup, which requires virtually no server resources.

Because early results in recent research are promising, we will use this idea for our research.

2.2.4 Charging for transmission of e-mail
An economic approach to restraining the transmission of spam is charging for the transmission of e-mail [9]. Charging makes it financially much less attractive for spammers to send vast amounts of e-mail and thus should theoretically limit the number of spam campaigns. Kraut et al. find that it causes senders to be more selective about what they send to who, but they also conclude that their design needs further development and testing. Most importantly, it is unknown how people should be charged for the e-mails they send.

We think charging could help, but there are disadvantages. Non-spammers would also have to pay for sending e-mail, this total cost could be larger than the current cost of spam. Spammers with sufficient financial resources would still be able to spam, and it also becomes more lucrative for them because of the fewer amount of spam campaigns the recipients will probably invest more time in reading the remaining spam campaigns.

We are looking for a technical approach, because we believe any non-technical approach needs to be supported by technical enforcement to be effective.

3. SCOPE AND REQUIREMENTS
Our literature study shows that we need to narrow the scope of our research to be able to develop an effective method to restrain the transmission of spam. Primarily because the LVS and HVS differ greatly in involved parties and used techniques, rendering a single solution impossible. Spam from HVS is already restrained quite effectively
through the use of blacklists \[13\]. Their e-mail servers are usually sending the spam with brute force methods and are therefore easily detected and listed.

Consequently, we will focus our research on the spam that is being sent by IVS. Our literature study shows that their spam is primarily transmitted through end-user hosts and hosting servers. The end-user is generally unaware of these issues, if we go a level higher in the network hierarchy we find the ISP's and hosting providers. They are suitable partitively to restrain the transmissions of spam: they have the required knowledge and skills to implement the required technology, and improving their network reputation is a great motivation for them. This will help them to avoid being listed in blacklists.

We developed the following set of requirements for an architecture to restrain transmissions of spam:

- **Must at least restrain 70% of the spam transmissions;**
- **Should require no more than 200 hours to implement;**
- **Should require no more than 5% of CPU, preferably no extra hardware required;**
- **Should not cause any disruptions to end-users currently using their e-mail services in a correct way.**

The implementation of new technologies such as IPv6 attests requirements such as low costs and easy implementation are crucial, because the parties involved are generally only motivated to implement new technology if they will experience advantages themselves, instead of merely creating advantages for others or the internet community as a whole. These requirements are also measurable and allow us to compare our proposed architecture.

4. **EXISTING SOLUTIONS**

Several partial solutions exist to restrain the transmissions of spam. To our knowledge, some large networks implement one or more of those solutions, but many smaller networks do not; this is supported by generated maps of the internet showing "bad neighborhoods" \[17\]. In this section we describe several of those solutions and rationalize why they do not conform to our requirements.

4.1 **Blocking the SMTP port**

The Simple Mail Transfer Protocol, used by all e-mail servers, runs on port 25. This protocol is used by e-mail servers to communicate with each other, but it is also used by end-users to communicate with their specific outgoing e-mail server. One could say that end-users are a very basic form of e-mail server, they only relay e-mail for themselves. This also means that end-users do not have to use an e-mail server from an ISP, they can deliver their e-mail instantly to the destination server should they choose so. Consequently, anyone can basically send e-mail to anyone without a way to restrict this, as displayed in Figure 4.1. One could say this is a major design mistake, but e-mail was invented over 40 years ago and in these times abuse was not really a concern. Since the internet is now much larger and used by the general public, abuse did become a problem.

When a computer of an end-users is compromised, the spammer installs software on the compromised host to let it act as a full e-mail server \[24\]. This means the compromised host will relay e-mail for the spammer and it consequently makes many connections over port 25 to different e-mail servers all over the world to deliver their spam. Therefore some ISP's decided to not allow any outgoing connections over port 25, they restrict SMTP connections to make sure only the ISP's e-mail server can be used to send e-mail.

![Figure 1: Possible routes to deliver an e-mail](image)

This is a very effective solution, but it does not solve the whole problem for ISP's. Not all spammers turn compromised hosts into full e-mail servers, they can also be commanded to use the ISP's e-mail server. It will cause an exponential increase of e-mail volume on the ISP's e-mail server, so they will need more hardware to process it. More importantly, many ISP's do not scan their outgoing e-mail for spam; by our experience it is too expensive due to the high amount of e-mails. One can confirm this by forwarding a spam e-mail, it will reach its destination and is not filtered by the ISP's outgoing e-mail server.

In this situation blacklist rejection policies can obviously not be used to limit the volume of e-mail, since all e-mail is coming from "trusted" end-user computers within the ISP's network. This means spammers basically have free passage through the ISP's e-mail server; it does not significantly restrain spam transmissions. Consequently, this solution only moves the problem to the ISP's e-mail server and is thus not a complete solution.

It also causes limitations for the end-users. Many users use several e-mail servers; for example, one to send e-mail from their private address and another one to send e-mail from their work address. Typically the e-mail server for the work address is outside the end-user ISP's network and since port 25 is blocked the end-user can not connect to his work's e-mail server. There is an extension to the mail protocol, providing a special submission port 587; however, this port is not used at large scale yet because e-mail clients by default connect through port 25.

4.2 **Rate limiting**

An effective but rather primitive approach is to simply limit the number of e-mails one end-user is allowed to send in a period of time. This can only be implemented by ISP's that also block the SMTP port for end-users, by hosting providers, and by webmail providers; a prominent example is GMail from Google \[5\]. GMail also limits the maximum amount of recipients for one e-mail. Typically, accounts that pass those limits will be temporarily suspended.
This only works against spam which is sent in bursts. However, research by Pathak et al. shows that spam campaigns go on for months and bots usually send only small amounts of spam through specific servers [16]. Even if this was not the case, bots would still be able to send small amounts of spam before being blocked, and many small chunks add up to a large spam campaign. Moreover, this approach is not discriminative, it also blocks legitimate large e-mail campaigns (for example, opt-in newsletters).

4.3 SpamAssassin
SpamAssassin is popular rule-based anti-spam software, adding and subtracting points based on the contents of an e-mail to determine whether it is spam or ham [14]. The major disadvantage of SpamAssassin is that it is quite resource-intensive [11, 17]. It is primarily used to filter incoming e-mail after the volume of e-mail has already been significantly cut down by using blacklist-based rejection policies.

SpamAssassin could also be used to scan outgoing e-mail. However, a small set of rules cannot be applied, because they are based on blacklists to check whether the source network can be trusted. If spamAssassin is used to scan outgoing e-mail, a lot of hardware is required to keep up with the volume of e-mail. Another disadvantage is that SpamAssassin does not perform well in specific cases; for example, SpamAssassin would most likely tag e-mails from a webshop selling Rolex watches as spam.

5. NEW ARCHITECTURE
5.1 Introduction of concept
Our literature study, overview of related work, and evaluation of existing solutions shows we need to focus on spam from low-volume spammers transmitted through compromised e-mail hosts and hosting accounts. System- and network maintainers of ISP’s and hosting providers have the requires skills and knowledge to be able to implement a new architecture. The major disadvantages of existing solutions are in the area of being obtrusive to the end-user, being ineffective to modern spam methods, and being expensive.

The new architecture we propose is specifically adjusted to the observations we made during our research to deal with all previously mentioned aspects. The novel idea is to use some of the existing solutions and concepts and combine them with the findings in recent research of Pras et al. [17] about the effectiveness of filtering by the use of core attributes of e-mails. This results in a central checking point with a simple filter, primarily based on cheap DNS lookups, which should have high accuracy in classifying outgoing e-mail as spam or ham at low cost. All outgoing e-mail will be forced through this checking point. If preferred, the architecture could be extended with SpamAssassin to scan each e-mail which has been classified as ham to achieve even higher accuracy; this does not require much additional hardware, because the amount of e-mail has already been significantly cut down by our new filter.

This new architecture requires only minor changes to the network’s infrastructure, but does not require any new additional hardware and can be implemented within a day. False positives should be extremely low, because it primarily enforces proper usage of the ISP’s or hosting provider’s e-mail server. This means this new architecture should not be obtrusive to end-users. It is also designed to catch modern methods of spam: low volumes of spam from many different hosts or hosting accounts. Besides the costs of implementing the architecture, it is basically free because DNS lookups require virtually no CPU, RAM or bandwidth. Furthermore, end-users generally do not need to make any changes in the settings of their client software.

The next paragraphs of this section describe the necessary adjustments that have to be made to the infrastructure, which core attributes are used for enforcing proper use of the infrastructure, and what a prototype in pseudo-code looks like.

5.2 Preparing an ISP’s network
We assume that an average ISP already owns some kind of basic transparent firewall and runs one or more e-mail servers. All outgoing e-mail should be forced through the e-mail servers of the ISP. Instead of blocking the SMTP port 25, we suggest to redirect the connection to a specific e-mail server which will give the end-user a 553 reject code, as described in RFC2821 [8]. This error should explain that the end-user should either use the ISP’s e-mail server or should connect through the submission port 587. Additionally, the error message could contain a link to an explanation about the reasoning behind the new network policy. This way the end-user has a clear understanding of what changed for what reason.

Alternatively, on Linux servers the ipt_owner module for the iptables firewall software can be configured to block all outgoing SMTP connections except for the applications which are specified by the administrator.

Next, all e-mail servers should be configured to forward their e-mail through the smarthost functionality which exists in every major e-mail server software. For an example Exim configuration, see section 5.3. In our architecture, the smarthost is the checking point. The already existing e-mail servers are transformed into so-called slave e-mail servers.

5.3 Preparing a hosting provider’s network
We assume a hosting provider’s network consists of servers which are all managed by the hosting provider. In case the

![Figure 2: Overview of architecture](image-url)
network also contains servers which are managed by end-users, instructions of section 5.2 should be applied and enforced to those servers.

Outgoing e-mail of all servers should be forced through the checking point. This is achieved through a technique that is called "smarthost". This means all the e-mail processed by an individual server is automatically forwarded to a central e-mail server, the smarthost, which is in our case the checking point. The individual servers are transformed into so-called slave e-mail servers. Configuring an e-mail server to forward to a smarthost is easy, an example configuration for Exim:

/etc/exim.conf:

```
smart_route:
  driver = manualroute
  domains = +!local_domains
  transport = remote_smtp
  route_list = + host.name.of.smart.host.server
  port = 26
```

For redundancy and reliability, multiple smarthosts can be configured. We forward all e-mail over port 26 (randomly chosen), which means that we can block (or redirect) outgoing connections over port 25 on all servers. This renders any malicious script that implements its own e-mail server useless, since they are unable to connect to any other e-mail server. If it is expected that hosting customers are using external e-mail servers in their scripts (which is highly unlikely), one could configure a simple e-mail server to which all blocked connections are redirected, in order to communicate a 553 reject code as described in RFC2821 [8] explaining what customers should do.

### 5.4 Setting up checking point

The checking point should be a smarthost which is highly optimized for processing many concurrent SMTP connections. It is beyond the scope of this paper to explain how to achieve optimal performance. A smarthost is a simple e-mail server which functions as a gateway to the slave e-mail servers; in our architecture the smarthost has the responsibility to check incoming e-mail for spam and relay the e-mail if it is classified as ham. E-mail should be dropped or rejected if it is spam. For the smarthost we prefer Postfix, as it is known for its high performance. We suggest to implement the filter, as specified in section 5.6, to be implemented as a daemon for optimal performance, but this is not required. Implementing the filter as a daemon saves a lot of CPU and RAM because it allows threading and the process is not fully restarted for every new e-mail. Because Postfix configuration for filtering with a daemon is complicated and because of limited space, this is an example for filtering by spawning a filter process for each to be filtered e-mail, as an example for filtering by spawning a filter program:

```
/etc/postfix/master.cf:

```
smtp inet n - - - - smtpd
  -o content_filter=CheckPoint

At the end of same file:

```
CheckPoint unix - n n - - pipe
  user=filter argv=/path/to/CheckProgram -f
  -e /usr/sbin/sendmail -oi -f ${sender} $recipient
```

Depending on the e-mail server and implementation method, a specific value has to be returned to the e-mail server to signal whether or not it should relay the e-mail. We suggest to initially reject spam with a 553 error code to be able to inform the end-user why their e-mail has been rejected, in case of a false positive.

### 5.5 Selecting core attributes

Restraining transmissions at the source network gives us a unique position in implementing a filter. Much more information is available, for example which domain names are used by the network. We want the filter to be light on resources, so we will limit our checks to DNS lookups and protocol checks. These lookups cost virtually no CPU and bandwidth. To be effective, we also limit our checks to so-called core attributes: elements in an e-mail which cannot be varied indefinitely.

We propose the following initial set of checks:

- Does the SPF record of the domain name used in the from-address designate an e-mail server in our network as allowed source?
- Are any of the URL’s mentioned in the e-mail listed at an URL blacklist or Google Safe Browsing?
- Does the used from-address exist?

Some newly introduced terms need explanation. Sender Policy Framework (SPF) is a relatively new DNS-based approach to authorize the use of a specific domain name for e-mail; one single DNS lookup allows us to check which e-mail server is allowed to send e-mail for a specific domain name. Unfortunately SPF also has the option to allow any e-mail server to send e-mail, in this case we should fall back to checking if www.domain.tld resolves to an IP within our network.

Next we scan the content of an e-mail for URL’s. We check them with several URL blacklists, which are also used by SpamAssassin: URIBL, SURBL, Day Old Bread, Spamhaus, Outblaze, and AbuseButler. These lists contain URL’s which are known or suspected to be used in spam and phishing activities. Each list enables site owners to easily and quickly remove listings of their website to prevent false positives. Spammers could also request removal, but this is futile since they are swiftly relisted because of continued malicious activities.

Google Safe Browsing [6] is an experimental service allowing developers to query Google’s lists containing sites suspected of malicious activities. Browsers such as Firefox are already able to use this service to prevent users from getting harmed by malicious websites. We specifically added this service to our filter, because we believe that many of the advertised websites in spam e-mails are hosted on compromised servers. Compromised hosting accounts are already used to send spam, so it is not unlikely they are also used to host the advertised websites. Furthermore, hacked websites are usually modified to infect visitors with a virus, so spammers can extend their botnet. Exactly those types of websites end up in Google’s automatically generated list.

Finally we check if the complete from-address is correct, this can be achieved by connecting to the e-mail server listed in the MX record of the domain name. We pretend to send an e-mail to the from-address; after issuing the RCPT TO command, the e-mail server will reveal whether the address exists or not by replying with a 250 or 550 code [8]. However, since this so-called sender callout can
be quite abusive [25], we advise to only perform this check if the previous two checks suggest the e-mail is ham. This way the check will only be performed on servers owned by the user of this architecture.

5.6 Prototype in pseudo-code
To specify a prototype we assume the filter has to return TRUE or FALSE, meaning SPAM or HAM. For the purpose of keeping the prototype simple, we assume each e-mail is delivered through a UNIX pipe and the result is returned to the standard output; therefore, no input/output handling code is included in this prototype.

```pseudocode
// Initialize variables and arrays $EMAIL = demime(get_input_from_pipe()); $FROM_ADDRESS = get_from_address($EMAIL); $FROM_DOMAIN = get_domain($FROM_ADDRESS); $SPF = get_spf_record($FROM_DOMAIN); $URLS[] = get_urls($EMAIL); $IP_SPACE[] = ...; // Array of network IP space

// Check SPF record IF ($SPF == any e-mail server) {
   // Fall back to A record $A = get_a_record(www.$DOMAIN); $SPF_OK = $IP_SPACE.contains($A);
} ELSE {
   $SPF_OK = $IP_SPACE.contains($SPF);
}

// Check all URL's, even if only one URL is // bad this test will return FALSE. FOR EACH $URL IN $URLS {
   $URL_OK = $URL_OK && in_blacklist($URL);
}

IF ($SPF_OK && $URL_OK) {
   // Only perform sender callout if other tests // check out return from_address_ok($FROM_ADDRESS);
} ELSE {
   return FALSE;
}
```

This pseudo-code shows that the filter is very simple and should be light on resources.

6. ANALYSIS
6.1 Introduction
In section 3, we have set several requirements for our new architecture. Primarily, the architecture has to significantly restrain spam transmissions, we have set a lower bound of 70%. In this section we will perform experiments to gain insight on how effective the architecture would be if it was to be implemented and used real-time. Furthermore, we will test the architecture for compliance to each of the remaining requirements.

6.2 Approach and used data sets
Dutch web hosting provider Antagonist B.V. has allowed us to perform experiments within their infrastructure. They have to deal with customers who were infected by a virus regularly, resulting in compromised hosting accounts; these hosting accounts are abused to transmit spam. Antagonist is therefore a typical company that could benefit from our architecture.

Due to privacy concerns, we are unable to perform experiments on real-time outgoing e-mail. To measure the actual effectiveness of our architecture, we would have to perform two steps: we let the checking point classify each outgoing e-mail, and then we manually inspect each e-mail again to determine if the classification was correct or not. The latter requires consent of Antagonist’s customers; due to limited time, the unlikelihood customers giving their consent, and the vast amount of customers it is unfortunately not feasible for us.

Instead, we initially opted to perform all our experiments on a data set compiled by Antagonist, containing so-called E-mail Feedback Reports (EFR’s) from US internet provider AOL. Every time a user of AOL identifies an e-mail as spam, AOL will automatically send an EFR to the originating provider (in this case Antagonist), allowing the provider to find and stop the source of the abuse. Every EFR contains a copy of the spam e-mail, including all headers; this would allow us to perform our tests, because we can be certain the EFR’s contain spam e-mails, eliminating the privacy issue. However, one of our tests is based on real-time URL blacklists. After an initial spam outbreak, there is usually a delay of a day before the first EFR’s come in. This means we can not reliably determine whether any URL’s were listed at the time the spam was transmitted, so this approach is not useful for URL blacklists. Nevertheless, we will use the data set of Antagonist to perform the remaining tests: the SPF check and the from-address check, these are not time related.

To gain insight on how effective the URL blacklists would be, we developed a different approach. All incoming e-mail of Antagonist is first filtered by several IP blacklists: BarracudaCentral, Spamhaus, SpamCop, PSBL, and NJABL. This ensures that any spam from high-volume spammers is rejected at the moment their e-mail servers connect to Antagonist. The remaining e-mail, which is either clean e-mail or spam from low-volume spammers, is scanned with SpamAssassin. SpamAssassin also performs several URL blacklist checks, and we have a log file containing information about each scanned e-mail. To measure URL blacklist effectiveness, we count how many of all e-mails, classified as spam by SpamAssassin, contain URL’s which were blacklisted at the time of scanning. This provides for a quite accurate indication of how effective these blacklists are. The delay between transmission and reception of a spam e-mail should be minimal, so this is the next best approach to performing experiments on actual real-time outgoing e-mail.

The list from Google Safe Browsing is not covered by SpamAssassin. GSB is not a URL blacklist based on DNS lookups, it is therefore difficult to add the test to SpamAssassin. To obtain an indication of its effectiveness, we will test the URL’s found in E-mail Feedback Reports. We can only determine if a URL was listed in the last 90 days, so the results are no more than an indication.

6.3 Measurements
Table 1 provides for an overview of the results extracted from SpamAssassin logs generated in the first week of January 2010. Please note that every customer who has enabled SpamAssassin is included in these statistics. An ‘URLBL hit’ constitutes a listing at one or more of the several URL blacklists that SpamAssassin is checking for. The overall effectiveness of URL blacklists is 74.4%.

Table 2 provides for an overview of the results extracted from E-mail Feedback Reports received from AOL in December 2009. The number of reports used to generate the statistics may appear low; however, not every spam e-mail transmitted to AOL is reported and not every spam e-mail
transmitted from Antagonist has its destination at AOL. Please note that these reports constitute a far larger number of actual spam e-mails, the exact number is unknown. The Google Safe Browsing hit is only an indicator: we could only determine whether or not the URL was listed in the last 90 days; therefore it is unknown if the URL was listed at the time of transmission.

Checking for SPF records and especially from-addresses appears to be very effective. The combined effectiveness of URL blacklists, SPF, from- addresses, and GSB is unknown, because the individual results have been extracted from different data sources and cannot be combined. We suspect, even higher hit percentages will be achieved when combining these checks.

### 6.4 Requirements compliance

In section 3 we have set several requirements for our architecture. Here we will discuss the compliance to each requirement.

**Must at least restrain 70% of the spam transmissions.** Individual tests such as the URL blacklist test and the from-address test achieved an effectiveness of over 70%. It is to be expected that combining the results of all tests (as proposed in the architecture) will achieve an even higher effectiveness.

**Should require no more than 200 hours to implement.** This paper provides for a specification of how to implement the proposed architecture. The configuration modifications that need to be made are trivial and should take no more than a few hours (including research for specific settings for the used e-mail server software). Currently, the checking point software has not been developed yet, although an experienced programmer should be able to implement it within a few days. We are considering to release an open source implementation; in this case, total adoption time will be limited to the time of installing a smarthost with the checking point software and applying the necessary configuration modifications at the existing e-mail servers, which takes definitely less than 200 hours.

**Should require no more than 5% of CPU, preferably no extra hardware required.** Since a working implementation of the checking point software does not exist, we are unable to measure this. However, a paper describing similar research by Pras et al. [17] states that their filter software, which only uses DNS lookups as well, shows an approximate increase of factor 20 in scanning throughput compared to SpamAssassin. Depending on the size of the infrastructure we estimate that the checking point software can coexist on one of the existing e-mail servers, or can run on just one additional server. However, for redundancy we advise to set up at least two checking point servers.

**Should not cause any disruptions to end-users currently using their e-mail services in a correct way.** Users who are currently not properly using the provider’s e-mail server will be informed through a 553 SMTP rejection error how to correctly transmit their e-mail. The risk of false positives should be extremely low. Antagonist hosts over 29,000 domains, and none of them are listed in URL blacklists, except for a low number of domains that are (correctly) listed at Google Safe Browsing. The SPF and from-address test should not cause false positives, because it is a simple protocol test and should never flag properly transmitted e-mail.

### 7. CONCLUSIONS

Spammers can be divided into two main groups: high-volume spammers (HVS) and low-volume spammers (LVS). Spam from HVS is already quite effectively rejected at the recipient through the use of blacklists. LVS transmit low volumes of spam through many compromised end-user hosts and hosting accounts, enabling them to transmit spam with IP’s that are usually not blacklisted yet. Therefore, our proposed architecture is designed to restrain spam transmissions from LVS.

ISP’s and hosting providers are suitable parties to implement such an architecture: they have the required knowledge and skills, and improving their network reputation is a great motivation for them. This will help them to avoid being listed in blacklists, allowing them to transmit e-mail for their customers reliably.

Our proposed architecture is easy to implement in an existing ISP infrastructure. Existing e-mail servers need to be modified to forward outgoing e-mail to a central checking point which performs several checks to classify e-mail as spam or ham. To keep costs low of such an architecture, we limited our filtering to the core attributes of an e-mail. Consequently, filtering software only needs to perform cheap DNS lookups or simple protocol checks, such as checking for SPF records, URL blacklist listings, and correctness of from-addresses.

Experimentation and analysis within the infrastructure of a Dutch web hosting provider shows that the individual checks have an effectiveness of 38% to 100%. Therefore, we expect a working implementation, which combines all checks, to be extremely effective. False positives are estimated to be very low, because the proposed checks merely enforce proper usage of e-mail services; this could not be confirmed due to privacy concerns.

### 8. FURTHER WORK

Further work should focus on implementing a working prototype to evaluate real-time performance and effectiveness. This requires a large amount of outgoing e-mail from end-users who allow their e-mail to be classified manually as spam or ham for research purposes.

We think the Google Safe Browsing check is very promising. During our research we were not able to perform...
real-time experiments to see if advertised URL’s in spam e-mails were listed at GSB as malicious. Nevertheless, our results from the E-mail Feedback Reports from AOL suggest there is a correlation between spam URL’s and GSB listing.

In ISP infrastructures, Netflow statistics could be a valuable information source for the checking point, to detect sudden bursts of outgoing e-mail from certain hosts. We were not able to experiment with this, because we only had access to a hosting provider’s infrastructure.

9. ACKNOWLEDGMENTS
We would like to thank the members of the "Broadband for All"-track for their valuable input and reviews. We would also like to thank Antagonist B.V. for allowing us to perform experiments and analysis within their infrastructure.

10. REFERENCES