The Effectiveness of Whitelisting: a User-Study

David Erickson
Stanford University
derickso@doemail.org

Martìn Casado
Stanford University
casado@doemail.org

Nick McKeown
Stanford University
nickm@doemail.org

Abstract

We believe this paper is the first extensive user-study of whitelisting email addresses. While whitelists are common in social networking and instant messaging (*e.g.*, buddy-lists), they have not caught on in email systems. Instead, most of us use spam filters that try to identify all the senders we do not want to accept email from. With whitelists we only need to identify the much smaller set of users who *can* legitimately send us email - generally most of these users are known to us. In our study we built and deployed a whitelisting email service - named DOEmail - along with a Thunderbird add-on to try and make whitelists easy to manage. During the past two years, over 120 users have used DOEmail. As expected, we find that almost no spam makes it to users' inboxes, and less than 1% of legitimate email is misclassified. We measure how hard it is for users to manage their whitelists, and conclude that after initial setup, the burden is very low. Our system uses challenge-response to identify whether unknown senders are legitimate. We give a quantitative evaluation of how effective challenge-response is, and the level of burden it places on the sender.

1 Motivation

Although current content-based spam prevention techniques have demonstrated some success in reducing the amount of spam reaching our inbox, the amount of spam sent is still growing, reportedly accounting for as much as 95% of all email in 2007 [8, 9].

A contributor to this rising tide is the notorious arms race between spammers and filter-writers. As filters improve, spammers strive to send as many permutations of their emails as possible until they find "winning" combinations that pass the spam filter checks. In turn, more stringent filters are deployed, leading to more legitimate email being tagged as spam (false positives). This seemingly endless spiral has created a cottage industry [1, 25] to help deliver marketing materials unfiltered to your inbox.

In the back of all our minds lies the fear that – in the long run – it is a losing proposition to try to identify all senders, and permutations of content, that we do not want in our inbox.

An alternate approach is whitelisting. A whitelist identifies the people you will accept email from – this includes our friends, family, and and other contacts. Clearly the size of the set of people we *want* to hear from is much smaller and more manageable than the set of people we *don't*.

However, whitelisting has not gained much traction as a way to reduce spam. The two most common concerns about whitelisting are [15, 16]:

- Whitelisting relies on authenticated source addresses. Historically, very few source addresses have been authenticated.
- Maintaining a whitelist is too much work for the user.

The first concern appears to be diminishing over time. A recent report [11] shows that, today, the majority (55%) of legitimate email uses at least one of the following source authentication schemes: SPF [29], SenderID [18], DomainKeys [14], DKIM [10]. In the long term we believe unauthenticated sources will cease to be a problem.

The second concern has received surprisingly little attention; we are not aware of any studies of the impact that whitelisting has on email senders and receivers. While it's clear that a whitelist requires maintenance and may filter legitimate mail, we are unaware of any reports of how the false positive and negative rates of whitelisting compare to content filtering in an operational whitelisting system.

In this paper we try to shed some light on the value and difficulty of whitelisting. We implemented and ran an operational whitelisting mail service, called DOEmail[1] for nearly two years supporting over 120 users. Here, we present the results of our user-study.

As one would expect, we find that whitelisting can protect our inbox much more effectively than content-

[1] DOEmail stands for Default Off Email.

based filtering. Further, the amount of day-to-day maintenance required by users was surprisingly low – on average less than half of the users performed any maintenance of their whitelists or blacklists each day. The burden on the sender was similarly low; fewer than 5% of senders had to manually add themselves to the receiver's whitelist by completing a challenge. False-positive rates were very low (0.81%), similar to effective content-based filtering. Since this value is non-zero, our users are still required to periodically scan through a list of non-whitelisted messages to identify legitimate email.

Our paper is organized as follows: section 2 provides an overview of our DOEmail system, section 3 contains statistics and analysis of the system's operation, section 4 describes our deployment experiences, section 5 includes a discussion of limitations of our system, section 6 explores existing work, and finally we conclude in section 7.

2 DOEmail

There are several existing email whitelisting services such as [19, 21, 22, 23, 4]; and in many ways DOEmail is similar to these. When designing DOEmail, our goal was to make it easy to use, and easy for us to collect usage data for this study. DOEmail is not particularly novel, and shares many features with the systems above. The benefit of implementing it ourselves was that we could tailor it for our research purposes, to collect the data discussed in the results section below.

2.1 Overview

DOEmail provides a basic forwarding service. Users sign up for a DOEmail account and redirect their existing email through it.[2] Each DOEmail user has a doemail.org email address (*e.g.*, `dericksо@doemail.org`), and their own whitelists and blacklists. Each user owns four whitelists:

- **Individuals.** Individual email addresses we will always accept email from, *e.g.*, `person@example.com`.
- **Domains.** Domains we will always accept email from, *e.g.*, `example.com`, or `.edu`.
- **Mailing lists.** Mailing lists we will always accept email from, *e.g.*, `interesting_list@lists.example.com`. As we'll see later, mailing lists need to be handled differently (and carefully).
- **Disposable addresses.** A list of addresses we can give to someone (or a web service) to reach us, regardless of the address they are sending from, *e.g.*, derickso+united_airlines@doemail.org; similar to those used in Mail Avenger [20]. Disposable addresses are easily revoked if a sender abuses them.

[2]Section 4.2 describes how this works in more detail.

Similarly, each user also owns two blacklists: one for individuals, and one for entire domains.

In DOEmail, whitelists can be populated in three ways:

1. **Manually by recipient.** Users can manually add entries to their whitelist. We describe the user interface below.
2. **Automatically add outgoing email addresses.** When we send email to someone, we are usually OK with them sending email to us. DOEmail allows users to automatically add all outgoing email addresses to the whitelist.
3. **Manually by sender.** If email is received from an address that doesn't match a whitelist or a blacklist, DOEmail sends a *challenge* to the sender. The challenge consists of clicking on a URL, taking the user to a website which (optionally)[3] contains a CAPTCHA[4]. When the sender completes the challenge, their email address is whitelisted and their email is delivered to the DOEmail user. The basic idea is that almost all SPAM is generated by computers; so SPAM-generating computers won't be able to get on our whitelist.

Our study examines many of the pros and cons of the last item on the list: the *challenge* that is sent to the sender. On one hand, we might expect that a CAPTCHA will raise the bar high enough to shut off almost all spam - and we'll see that it does. On the other hand, it may raise the bar too high for some senders. Our friends, family and colleagues might be put-off, offended or confused when we ask them to fill out a CAPTCHA, or they might be unable to do so (*e.g.*, the visually impaired). Our study tries to scientifically evaluate this tradeoff.

In summary, figure 1 shows the decision process DOEmail follows when an email arrives. First, DOEmail checks the blacklist; if it matches, the email is dropped.

Second, DOEmail checks to see if the email comes from another DOEmail server. There is a danger that two DOEmail users never whitelist each other because the *challenge* never reaches their inbox. Therefore, outgoing DOEmail challenges are marked,[5] so the challenge bypasses all checks and is delivered directly to the inbox.

[3]DOEmail users can specify whether new senders must complete a CAPTCHA or not

[4]Essentially a distorted image that is easily recognizable by humans, but not by computers [27]. Currently DOEmail only supports reCAPTCHA [3] which contains a visual and optional audio-based turing test, but there is no reason this approach cannot be extended to other tests.

[5]The mark is a X-DOEmail-sig email header containing an RSA encrypted hash covering the body of the message and a timestamp.

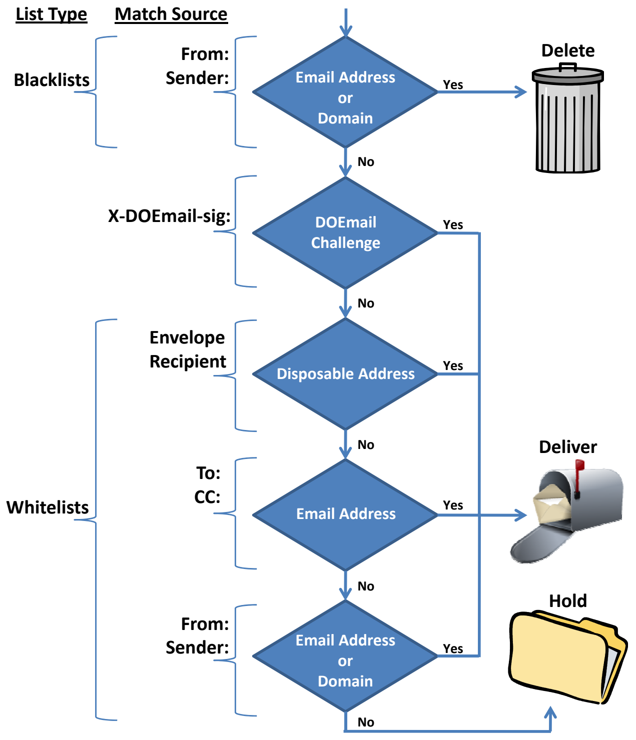


Figure 1: Decision Flow Diagram

Third, DOEmail checks to see if the email matches any of the recipient's four whitelists. If it does, it is delivered to the inbox.

Finally, if the email doesn't match a blacklist or whitelist, it is added to the user's *pending list*, and a *challenge* is sent to the sender. If the sender correctly completes the challenge, the email is removed from the pending list and delivered to the inbox; the sender's email address is added to the user's whitelist.

2.2 Challenges and pending email

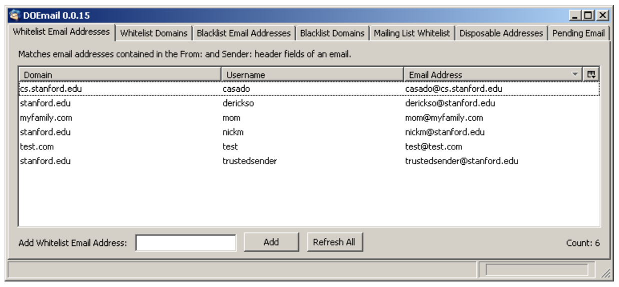
The challenge allows a human sender to add themselves to the recipient's whitelist, but makes it difficult (and not cost effective) for a spammer. While a spammer could perform the required functions to satisfy the challenge, it is unlikely they will take the effort to do so. Of the over 500,000 emails DOEmail processed, we only verified this happening twice, both times from fraudulent Nigerian-based scams [24].

Similar to a spam folder, the user is able to view the contents of the pending email folder and can manually whitelist (or blacklist) email addresses and domains associated with waiting messages.

DOEmail holds pending email for up to three weeks from the receive date, and then deletes it.

2.3 User Interface

When designing DOEmail, our main goal was to make it easy and appealing enough to use that we could



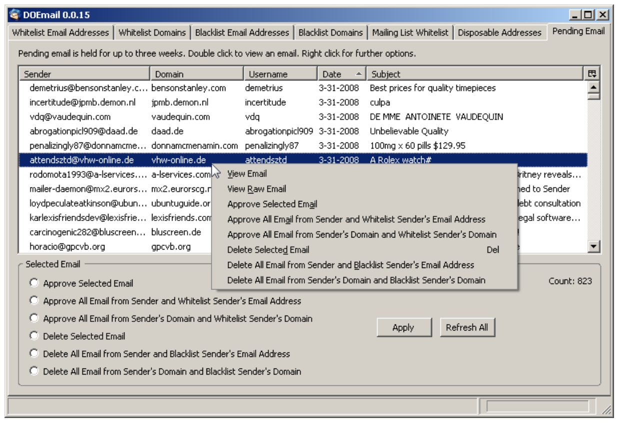
Matches email addresses contained in the From: and Sender: header fields of an email.

Domain	Username	Email Address
cs.stanford.edu	casado	casado@cs.stanford.edu
stanford.edu	derickso	derickso@stanford.edu
myfamily.com	mom	mom@myfamily.com
stanford.edu	nickm	nickm@stanford.edu
test.com	test	test@test.com
stanford.edu	trustedsender	trustedsender@stanford.edu

Add Whitelist Email Address:

Count: 6

(a) Email Whitelist


DOEmail 0.0.15
Whitelist Email Addresses
Whitelist Domains
Blacklist Email Addresses
Blacklist Domains
Mailing List Whitelist
Disposable Addresses
Pending Email
View Email
View Raw Email
Approve Selected Email
Approve All Email from Sender and Whitelist Sender's Email Address
Approve All Email from Sender's Domain and Whitelist Sender's Domain
Delete Selected Email Del
Delete All Email from Sender and Blacklist Sender's Email Address
Delete All Email from Sender's Domain and Blacklist Sender's Domain
Apply
Refresh All

Pending email is held for up to three weeks. Double click to view an email. Right click for further options.

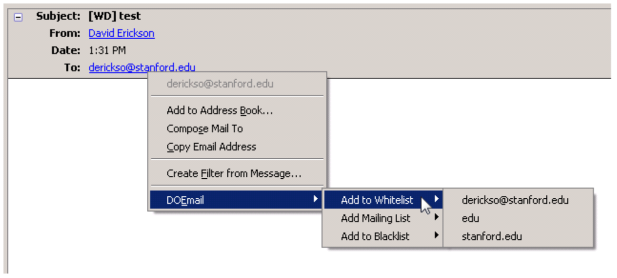
Sender	Domain	Username	Date	Subject
demetrius@bensonstanley.c...	bensonstanley.com	demetrius	3-31-2008	Best prices for quality timepieces
incertitude@jpmb.demon.nl	jpmb.demon.nl	incertitude	3-31-2008	culpa
vdq@vaudequin.com	vaudequin.com	vdq	3-31-2008	DE MME ANTOINETE VAUDEQUIN
abrogationpicl909@daad.de	daad.de	abrogationpicl909	3-31-2008	Unbelievable Quality
penalizingly87@donnamcme...	donnamcmenamin.com	penalizingly87	3-31-2008	100mg x 60 pills $129.95
attendsztd@vhw-online.de	vhw-online.de	attendsztd	3-31-2008	A Rolex watch#
rodomota1993@a-lservices....	a-lservices.com			ritney reveals...
mailer-daemon@mx2.eurors...	mx2.eurorscg.r			ned to Sender
loydpeculateatkinson@ubun...	ubuntuguide.or			lebt consultation
karlexisfriendsdev@lexisfrie...	lexisfriends.com			egal software...
carcinogenic282@bluscreen...	bluscreen.de			!
horacio@gpcvb.org	gpcvb.org			

Selected Email

- Approve Selected Email
- Approve All Email from Sender and Whitelist Sender's Email Address
- Approve All Email from Sender's Domain and Whitelist Sender's Domain
- Delete Selected Email
- Delete All Email from Sender and Blacklist Sender's Email Address
- Delete All Email from Sender's Domain and Blacklist Sender's Domain

Count: 823

(b) Pending Email



Subject: [WD] test
From: David Erickson
Date: 1:31 PM
To: derickso@stanford.edu

(c) Dynamic Context Menus

Figure 2: DOEmail Thunderbird add-on

encourage 100-200 users to participate in our study. We soon realized that DOEmail needed an easy-to-use user interface. Building good, intuitive user interfaces is not easy and consumed a large part of our study.

We built two user interfaces to help users manage their whitelists, blacklists and pending email. The first is a Mozilla Thunderbird add-on, the second a web-based interface.

2.3.1 Thunderbird add-on

Thunderbird is a powerful, free, and widely-used email client, and was used by 44% of our users.

The Thunderbird add-on has too many features to list here, but the main ones are:

1. **Whitelist and blacklist management.** Figure 2(a) shows the add-on's main page. Each tab contains a specific whitelist or blacklist and an interface to add/remove from it. Also, any email ad-

dress visible in Thunderbird can be quickly added to a whitelist or blacklist by right-clicking on it (see figure 2(c)). This feature has proven to be very popular and easy to use.

2. **Pending email folder.** The pending email tab (see figure 2(b)) displays a list of the user's pending email (*i.e.*, email for which a challenge has been sent, but not yet completed). The user can whitelist (or blacklist) the sender or their domain, and move the email to their inbox. Or the user can manually delete the email. To help the user decide, they can open and view the email in a Thunderbird window.

3. **Mass import of addresses to whitelist or blacklist.** Users can import all (or a subset of) the email addresses found in a Thunderbird folder (inbox, sent, etc.) into a whitelist or blacklist. This feature is very useful for first-time DOEmail users.

4. **Processing result.** A column containing a graphic representing the type of rule that each email matched during DOEmail's processing is displayed in Thunderbird's main email list.

We tried to design and test all these features carefully so-as to make it easy for new users to migrate to DOEmail, while showing them the control and power they have over their whitelist and blacklist.

2.3.2 Web Interface

The web interface provides similar functionality to the Thunderbird add-on. Users can edit their whitelists and blacklists, and view their pending mail folder. Users can also personalize the outgoing *challenge* email, and ask for a daily summary of pending email.

The web interface displays a variety of usage statistics and graphs, including: the total number of email received by the user per day, the number of emails that matched whitelists and blacklists, how many matched neither and went into the pending folder, the number of CAPTCHAs completed by senders, and the number of manually confirmed emails by the recipient.

2.4 System Setup

Figure 3 is a high-level overview of the DOEmail system. Users forward email from their existing accounts to their DOEmail address. Incoming mail is handled by Mail Avenger which acts as the incoming SMTP server. Received emails are post-processed by a number of Python scripts which perform the blacklist and whitelist filtering, store pending email locally, and forward accepted email out to user accounts. Integrating with existing email accounts requires processing to determine whether or not incoming email has already been passed through DOEmail.[6]

[6]We found this to be a non-trivial hurdle when migrating new users which we discuss further in section 4.2.

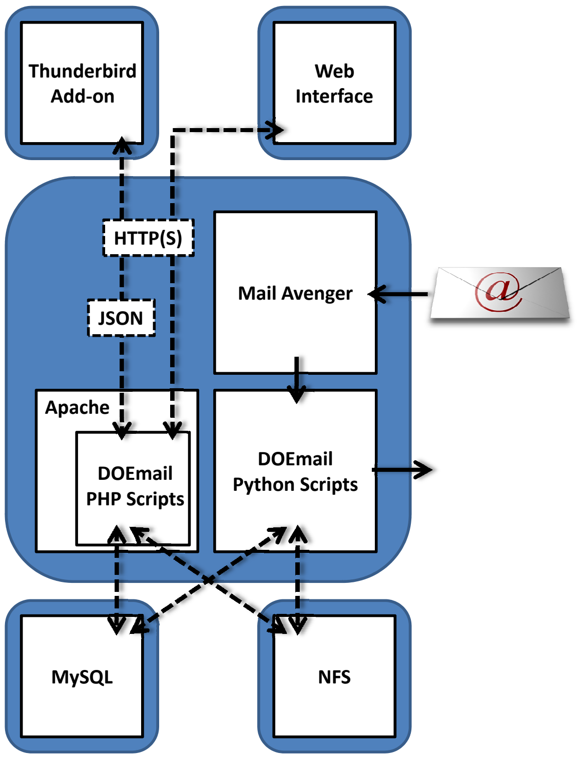


Figure 3: System diagram showing how mail flows in and out of DOEmail denoted by solid lines, as well as inter-system communication denoted by dashed lines. Dark squares with rounded corners denote single physical systems.

The Thunderbird add-on is based on JavaScript and XUL [7], and communicates with a PHP service page using HTTPS, and transmits data encoded in JavaScript Object Notation (JSON) [2]. The DOEmail servers run an Apache web server with access to the DOEmail configuration state, the pending email list, and the whitelists and blacklists. The web interface is built using PHP.

Incoming email is handled by a single DOEmail server with redundant backup servers for fail-over. If the primary server goes down, a backup server is automatically allocated the active IP. Account information and all whitelists and blacklists are stored in a MySQL database. Pending emails are stored on a filesystem with snapshot support.

2.5 DOEmail Deployment

DOEmail has been operational at Stanford University for nearly two years and has hosted over 120 total users. We deployed DOEmail in the Electrical Engineering and Computer Science departments at Stanford University. Many professors, staff and students actively use it on three large departmental and university email servers, including `stanford.edu`. Nearly 70 of DOEmail's users were from off campus, forwarding their email from a variety of email servers through DOEmail.

DOEmail currently receives around 9,000 emails per day during the week, of which roughly half do not have a corresponding whitelist or blacklist and are thus held pending approval. On average, the system receives 60 challenge responses from senders per day.

3 Results

We collected a large quantity of data with DOEmail from real users, and we try to make sense of it here. As we'll see, it's a little hard to make "apples with apples" comparisons, because we have no definitive test of whether an email is really spam or not. However to get an understanding of our false negative rate we modified the Thunderbird add-on to explicitly ask the user if they consider an email spam.

3.1 Our Data

The data reported in this section was collected from 07/13/07 through 02/29/08, encompassing a total of 592,794 emails received by 112 user accounts. An overview of the data is presented in table 1.

DOEmail logs the headers of all incoming email, all manual edits of whitelists and blacklists, and events involving sender challenges and responses. For comparison, we ran all incoming email through SpamAssassin [5][7].

DOEmail divides email into three categories: accepted, deleted, and outstanding. "Accepted" emails are those that matched a whitelist entry, were confirmed by the sender through challenge-response, or confirmed manually by the DOEmail user. 55.55% of email was "accepted".

Our system deleted just 37.84% of received email – much lower than the 90+% spam reported elsewhere [8, 9]. This is for three reasons. First, users forward email from other systems (*e.g.*, `stanford.edu`, or Gmail), and these sytems already filter some email that is marked as "very high likelihood" of being spam.[8] Second, many of our users are students and have recently created email addresses. It takes time for an email address to be publicized and end up in spam lists. Last, several of our users are system administrators who receive large quantities of (non-spam) auto-generated reports from systems they manage.

Email is "Deleted" for one of three reasons: if it is blacklisted, if it expires while on the pending list (the default timeout is three weeks), or manually deleted from the pending list by the user.

When the experiment ended, 6.61% of received email was still sitting in users' pending folders.

[7]A very popular content-based spam filter.

[8]This does have the interesting consequence that DOEmail receives spam that could be considered "the worst of the worst", spam that has already made it through other filters, and yet, is responsible for stopping it from reaching the user.

	# of emails	% of total
Total Email	592794	
Accepted	329277	55.55
Whitelist	320097	53.40
Sender Confirmed	4382	0.74
Manually Confirmed	4798	0.81
Deleted	224320	37.84
Blacklist	84065	14.18
Expired	79297	13.35
Manually Deleted	60958	10.28
Outstanding	39197	6.61

Table 1: Number of emails received between 07/13/07 and 02/29/08 categorized by the action taken by DOEmail

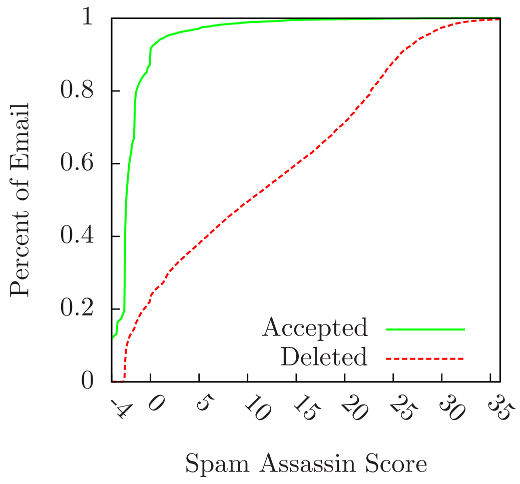


Figure 4: CDF plots of the SpamAssassin scores of accepted and deleted email.

3.2 False positives

A false positive means an email was rejected by DOEmail, but should have been accepted. We have no definitive measure of whether or not an email should have been accepted, but we can estimate it. If a user manually accepts an email in their pending folder, we assume it should have been accepted and was therefore a false positive. Our users manually accepted 0.81% of the total received email, which was 1.46% of accepted email.

This is an underestimate of the real false-positive rate: a user might not check the pending file and miss an email that should have been accepted. However, we noticed that - particularly when they were new to DOEmail - most users checked their pending file quite regularly.

3.3 Comparing with SpamAssassin

SpamAssassin calculates a score for each email — the higher the score, the more likely it is spam. If it crosses a threshold it is dropped. We were interested to know how SpamAssassin would score the email that DOEmail accepted or deleted. Figure 4 shows the CDF of these SpamAssassin scores.

First look at the "Accepted" curve. As expected, email that makes it to the inbox also has a very low SpamAssassin score.

The "Deleted" curve tells a very different story: almost all scores in the range 0-30 are equally likely, showing there is no good threshold for SpamAssassin to use. For example, if we filter all emails with a score greater than 2, we will accept 90% of legitimate messages, but accept nearly 30% of spam. If we want to only get 10% of the spam (SA score of -2.4), we only receive 54% of the legitimate emails. This quite surprising result illustrating the extent of the continued arms race between spam generators and spam filter creators.

To estimate the false negative rate of DOEmail, we instrumented the Thunderbird add-on to track the deletion of email that had passed through DOEmail, randomly querying the user (at most once per day) if the email being deleted is considered spam. Of 198 queries, 29 (14.64%) were reported as being spam. During the sampling period we tracked 17,779 deletions. The users who were queried received 248,283 emails during the timeframe of the email they deleted. If we assume that these users deleted all their spam during this sampling period, and that 14.64% of all deletions were spam, then we have approximately 2603 spams that got through. This results in an estimated 1.05% false negative rate. In practice we expect this rate to be even lower; of the email that users identified as being spam, 58.62% of it arrived through a course-grained whitelist domain rule. These could likely all be eliminated by using only whitelist email rules.

DOEmail also makes it very clear that - even to human users - the definition of spam is not always precise. Is a chain letter from a friend spam or just a slight nuisance? When we receive five identical announcements for the same event (*e.g.*, a conference or a talk), are four of them spam? If we are not interested in the event, should all five copies have been identified as spam? The answer is not obvious. Another example of hard-to-classify email - familiar to any faculty member - is the email received from prospective students around the world just prior to a university admissions deadline. If the same email is sent individually to every faculty member, most professors would describe it as spam. If an email is sent to just one faculty member, perhaps it was not spam. Clearly, the definition of spam is context specific. In this particular example we noticed that DOEmail's challenges had the interesting and desirable effect of slightly "raising the bar" for the sender; and in general only the legitimate senders took the time to respond to the challenge.

3.3.1 User Overhead

Managing whitelists We wanted to understand the burden DOEmail imposes on its users – how much work it takes to maintain whitelists and blacklists. Figure 5 shows the average number of times users modified their whitelists and blacklists per day. An action here is defined as adding an entry to, or removing one from a list. The user interface allows users to add and

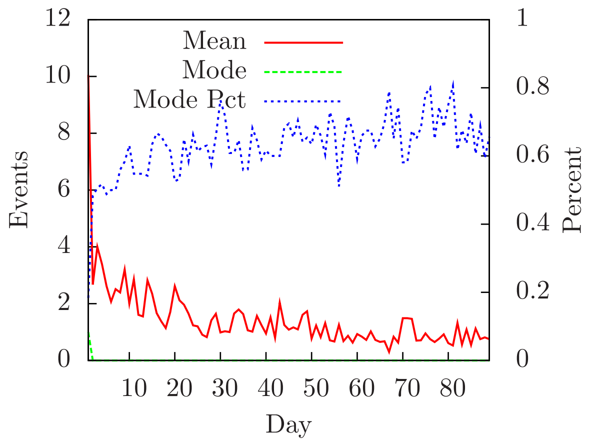


Figure 5: Shows the median, mode, and what percentage of users the mode accounts for, of events performed by a user per day after registering for DOEmail

remove multiple entries at a time from whitelists and blacklists; this graph considers such operations as a single action. We evaluate users starting from the day they sign up until they cease being active, or hit the 90 day activity mark.

As expected, maintenance is highest to start with - when users are setting up and tuning their whitelists and blacklists. It is interesting how quickly the maintenance drops. On the first day of use, the average number of actions performed is 10. This decreases to half by the second day. After three weeks users are interacting with the system less than twice per day, and eventually to just once on average.

We noticed that a minority of users took a very active role in managing their mail accounts and their interactions inflate the results. On the other hand, the mode indicates that the majority of users don't interact with DOEmail at all on a day-to-day basis. As is shown in the graph, after the first day, over 50% of the users don't perform any manual modifications to their whitelists, and this number increases to greater than 60% over the first two months.

Challenge emails and the pending folder We also wanted to know how much effort users devoted to managing the pending folder, and how effective the challenge-response emails are.

Table 1 shows that 1.55% of email was accepted, but required action by either the sender or the DOEmail user. If DOEmail *did not* send challenges, this is the amount of email users would need to manually confirm.

With *challenge* emails, our users manually confirmed 0.81% of accepted email; *i.e.*, challenge emails roughly halved the number of manual confirmations. We expected the challenge emails to reduce the amount of manual work by more than this.

Being users of the system ourselves we posited that much of the manually confirmed email consisted of auto-generated email, such as receipts, newsletters,

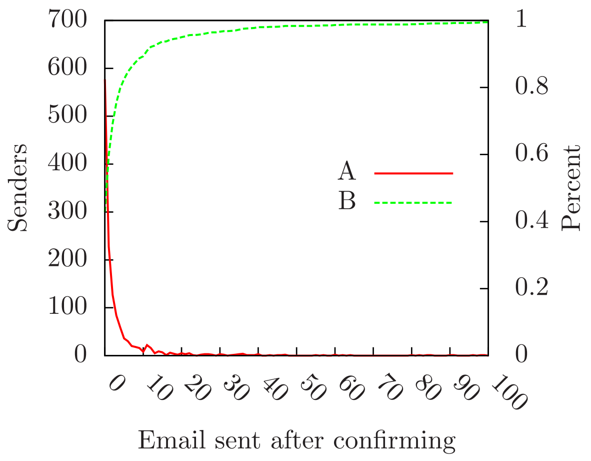


Figure 6: The number of senders sending specific quantities of email to a DOEmail recipient for 90 days after completing the challenge response process, along with the cumulative percentage

etc. Challenges sent in response to such email would almost never reach human eyes, and thus never be confirmed. We instrumented the manual email confirmation process to randomly ask DOEmail users if the email they are confirming appeared to be auto-generated. 160 responses were received, 104 of them (65%) reported the email being confirmed was auto-generated. Using this result with the data in Table 1 we learn that 72% of legitimate senders *do* reply to the challenge, confirming their email. Factors contributing to the 28% of senders that do not reply may include: challenges being filtered as spam before reaching the original sender, the technical hurdle for new email users, or distrust of email containing links. To try and mitigate distrust of challenges, DOEmail users can personalize them, for example by adding a photograph of the user.

Because such a large percentage of email requiring manual intervention by the DOEmail user was auto-generated, we are exploring ways to make it easier for users to whitelist these email addresses when they sign up for an online service; for example, by adding DOEmail support to Firefox. This in-turn will require willing disclosure of email addresses that website operators use to send such email.

Blacklists We were quite surprised that users blacklisted 14.18% of their received email. But on closer inspection we found that 96.4% of all blacklisted email was for one user who was on many unwanted mailing lists. From the user's perspective, blacklisting email addresses and domains to eliminate this email was faster than hunting down and attempting to unsubscribe from all of the disparate sending locations.

If we remove this user from consideration, the remaining users blacklisted less than 1% of their received email.

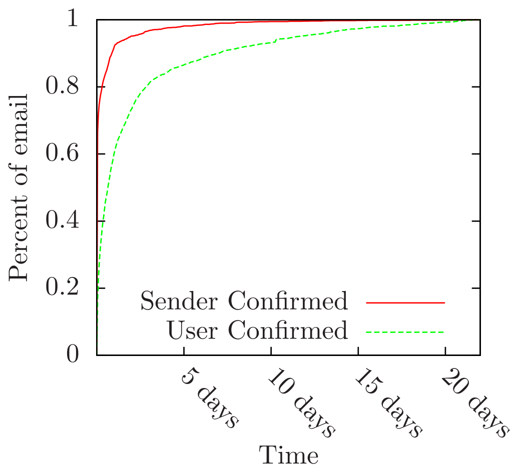


Figure 7: CDF plotting how long it takes senders to complete the challenge-response process compared with DOEmail users manually confirming email

3.4 Sender Overhead

It is hard to quantify how painful or burdensome DOEmail challenges are. Anecdotally, we heard many responses — some senders don't mind, and are happy to fill out the challenge (once). Some senders simply ignored it. Some were offended to be asked to verify they are humans.

For those senders who completed challenges, we tracked the number of subsequent emails they sent to the same receiver over a 90 day period after the initial email.

The line labeled A in figure 6 shows a curve of the number of senders (left y-axis) and the quantity of email sent (x-axis). The line labeled B is a cumulative percentage of the same values. 43% of senders did not send a single email to the DOEmail user after the initial email. And 75% of senders sent 3 or fewer emails after confirming. For this group the challenges make up a relatively high percentage of the total communication (over 25%).

But perhaps we shouldn't be surprised – could it mean that users are very good at creating whitelists that cover almost everyone they communicate with frequently? This indeed appears to be the case: only 4.4% of all user/sender pairs who exchanged valid email completed a challenge response. This is encouraging news for whitelisting. It appears that many users whitelist entire trusted domains (*e.g.*, `stanford.edu` or `my_employer.com`), drastically reducing the number of challenges that need to be sent, but putting them at risk of receiving more spam.

3.4.1 Response times

We also explore the distribution of time it takes senders to respond to the challenges. In figure 7 we plot a CDF of the delivery times of email from unknown senders that are eventually confirmed by the sender, or manually confirmed by the user. Over half of the email confirmed by senders occurs in under 15 minutes from the time it was initially delivered to the system, and 95% within the first 24 hours. As expected, the manually confirmed curve is delayed to give the sender time to confirm. However, email is confirmed by DOEmail users as long as the 3 week expiration date, suggesting that retention beyond 3 weeks may be necessary.

4 Deployment Experience

We faced many practical challenges and unforeseen complications while running the operational system. We describe some of them here.

4.1 Mailing lists

Mailing lists created an interesting problem. DOEmail maintains a separate whitelist for mailing lists; instead of matching on the sender's email address, it matches on the `To` and `CC` fields. This is because emails from mailing lists typically set the `To` or `CC` field equal to the name of the mailing list. If the user is subscribed to the mailing list, it is assumed that all emails from the list should be accepted (unless the sender is blacklisted).

Alternatively, we could have whitelisted the List-ID header (recommended by RFC 2919 [13]), but many mailing lists don't yet adhere to the RFC that calls for the List-ID email header.

DOEmail assists users in identifying mailing lists using the following algorithm. If an incoming email doesn't match a blacklist or whitelist, check to see if it contains List-* headers. If it does then extract all email addresses contained in the `From`, `Sender`, `To`, and `CC` header fields. Check if any of these email addresses are contained in the List-* headers. If a match is found, assume it is a mailing list address based on the observation that some mailing lists place their identifiable address into one of the List-* headers. We keep track of all the email addresses that match the List-* header, as well as addresses from emails with no match. The addresses seen most frequently will likely be mailing lists.

Potential mailing list addresses are presented to the user through the web interface. They can then be added to the whitelist, deleted, or ignored.

To avoid creating spam of its own, DOEmail does not send challenge emails to mailing lists.

4.2 Account integration

To attract as broad a user base as possible, DOEmail was designed to integrate with existing email accounts. This proved to be harder than we anticipated. Novice users had a difficult time configuring their existing email accounts to forward email to DOEmail. Of those that were able to forward, some had mail servers which would rewrite the `From` or `Sender` email header of all forwarded email with the user's email address, behavior that is consistent with forwarding from a mail client, but not a server.

The ability to re-inject DOEmail filtered email back into the user's original email account turned out to be prohibitively difficult. Supporting this required the user's server to differentiate between email received for the first time, and email that has been processed by DOEmail. We have been unable to identify a free major email provider that can enable this in a clean way[9].

At Stanford this turned out to be much easier. For example, the main stanford.edu domain forwards email to a final destination server, which enabled us to interpose DOEmail between the two. On other Stanford systems we used specially configured email addresses (eg. derickso+clean@example.com) that when sent to, would deliver cleaned email directly to the inbox, meanwhile all other email is forwarded to DOEmail. As a last resort it is possible to use two email accounts, one for receiving and forwarding to DOEmail, and one to receive all the clean email, although this is a cumbersome solution.

4.3 Sender challenges

We had to craft challenges so that they wouldn't be tagged or dropped by the sender's spam filter. We implemented multiple techniques to reduce the chance this would happen. Our outgoing challenges are composed of a text-only section of the email, along with an optional (on by default) HTML portion. We embedded the images in the email itself to decrease the likelihood that the email would be considered spam due to linking to external images. We also created an SPF entry for our domain, specifying which IP addresses are approved for sending email from doemail.org. Lastly we hash and sign all outgoing challenges with DomainKeys signatures. Despite this it appears some major providers (*e.g.*, Yahoo!) classify our challenges as spam.

5 Limitations

Backscatter Challenges sent in reply to spam containing forged sender addresses are considered "backscatter". Backscatter is often considered as contributing to the spam problem because it can be sent to users that never originated an email, and can also cause a flurry of failed delivery status notifications to be sent if the email address was invalid.

We believe this problem can be overcome through wide-spread adoption of sender authentication. If a domain advertises its use of sender authentication, and an incoming email purports to be from the domain,

[9]We were able to do this with Google's Gmail, however it involved inserting a character string into each email's subject that could 'tag' it as having been through DOEmail, however we felt this was too intrusive a solution to be viable.

but cannot be verified, then the system will not send a challenge email to the spoofed sender address. Thus it is in the domain administrator's best interests to deploy sender authentication to protect themselves from backscatter.

Automated senders Email sent from automated senders such as registration confirmations or newsletters cannot be confirmed through challenge-response. As mentioned earlier, DOEmail supports "disposable" addresses that could be used when the sender's address is unknown. Other methods of handling automated senders involve pre-whitelisting an entire domain, or waiting for the email to arrive and manually confirming it from the pending email.

Going forward we have high hopes that as whitelisting becomes more common, websites will be more explicit in listing the email addresses they plan to send to you from, allowing users to pre-whitelist the addresses. Since beginning work on this project we have seen a large increase in the frequency of such disclosures.

Challenge collision Multiple users behind different whitelisting services can complicate the use of challenges. In the worst case, the original sender of the email will have to comb their pending list for the challenge sent by the recipient[10]. One approach to addressing this is for whitelisting services to standardize around sending challenges from a known address or domain (with source authentication). This way, senders can auto-whitelist the address or domain when sending email, in expectation of receiving a challenge. Another approach is to create a challenge-response authority which provides the credentials to allow senders to sign "legitimate" challenges and for receivers to verify them. We don't feel either of these approaches are particularly elegant and believe this to be a interesting area for future work.

Mechanical turk attacks In the context of this paper, a mechanical turk attack would strive to harness large numbers of users to solve the CAPTCHAs presented by the whitelisting system. Generally the users doing the work are "paid" via access to online content. While this may pose a real threat for CAPTCHAs used to protect sending accounts (from which a spammer may send millions of messages), we don't believe it is a realistic attack to gain access to the inbox of a single user. Because users may immediately blacklist the offending address, the cost to the attackers converges on answering one CAPTCHA per email.

6 Related work

There have been numerous proposals for combating spam (and a relative paucity of user based studies). We briefly discuss related mechanisms here.

[10]Note that there isn't danger of a loop as challenges are sent from addresses that will not issue a reply

Leveraging existing social networks A number of projects have proposed leveraging existing social networks to help create email whitelists. Re: [15] proposes a privacy preserving approach to using friend-of-friend social networks to expand a single user's whitelist. Johansen *et al.* [17] examine algorithms for identifying communities of interest (akin to social networks) from existing email traffic patterns. This can be used to aid in whitelist generation or identify networks likely to send spam. In [12], Boykin *et al.* use graph-theoretic methods to analyze email headers. The resulting data is used to create white, black, and gray lists based on the characteristics of existing email networks.

Sender reputation Taylor [26] explores the use of sender reputation in the Gmail email service. Sender reputation allows Gmail to classify most senders as good or bad, remaining email is run through traditional statistical classification. Sender reputation could provide a useful input to user (or administrator) automated white and blacklist entries.

Changing the cost model for senders Another class of proposals attempts to alter the current cost model of sending spam in which the marginal overhead of sending an additional email converges on zero. The Penny Black Project [6] suggests charging a small fee for every email. This would not effect low volume senders such as users, but could greatly impact spammers who rely on large volumes (and consequently list managers as well unless special measures are taken). Walfish *et al.* [28] propose limited quotas per sender and offers a scalable mechanism for enforcing it.

Challenge-Response for whitelisting Many open source and commercial anti-spam systems contain elements found in DOEmail such as whitelists, blacklists, and challenge-response. Examples from the open source community include, TMDA (Tagged Message Delivery Agent) [19], Active Spam Killer (ASK) [21], and Qconfirm [22]. There are also a number of commercial offerings such as Spam Arrest [4] and Clean My Mailbox [23].

7 Conclusions

In the past, whitelisting probably didn't make sense, because it was so easy to spoof sending addresses. Nowadays, spoofing is much harder to do - and will continue to be so - making whitelisting an interesting option. Our study suggests that whitelisting is very effective, and significantly outperforms SpamAssassin in both false negatives and false positives. It is interesting to note that this is achievable on a simple prototype system with significantly less engineering effort than is devoted to creation of spam filters. But this shouldn't be surprising: like a buddy-list in IM, a whitelist tries to precisely identify the people we communicate with, or who we allow to send us email. Unless we make a mistake, we will not allow a spammer to send us email. We should expect a well-engineered whitelisting email service to behave almost perfectly.

But the controversy with whitelisting is not the performance, but the overhead imposed on the users. In our system, both the sender and the receiver are involved in maintaining the user's whitelists. For senders, our analysis shows that less than 5% of sender-receiver pairs ever went through the challenge-response process. Our data indicates that the burden on the user (the receiver of email) is also very low, with most users doing no manual maintenance after the first few days. Further, less than 1% of incoming email required manual confirmation, 65% of which originated from automated senders. This can be optimized through a combination of a better user interface, allowing users to whitelist email addresses when they register for an online service, and more forthcoming webmasters. We believe that a good user interface is the key to low maintenance; our system, implemented by a single engineer, placed very little burden on the user. In the future, we expect that finely-tuned whitelisting systems (based on more experience) will require even less maintenance.

In summary, this experiment has demonstrated that a whitelisting-based system can be both highly effective and low maintenance, challenging commonly held beliefs about such systems. We believe that whitelisting deserves much closer attention than it has received in the past.

Acknowledgments

We thank the anonymous reviewers of this paper for their comments and suggestions, and all the users of DOEmail that made this research possible. This work was supported by the Stanford Clean Slate Program.

References

[1] Certified Senders Alliance. http://www.certified-senders.eu/.

[2] JSON: JavaScript Object Notation. http://www.json.org/.

[3] reCAPTCHA. http://recaptcha.net/.

[4] Spam Arrest. http://www.spamarrest.com.

[5] SpamAssassin. http://spamassassin.apache.org/.

[6] The Penny Black Project. http://research.microsoft.com/research/sv/PennyBlack/.

[7] XML User Interface Language (XUL). http://www.mozilla.org/projects/xul/.

[8] Barracuda Networks. Annual Spam Report. http://www.barracudanetworks.com/ns/news_and_events/index.php?nid=232, December 2007.

[9] Commtouch Software. Q4 2007 Email Threats Trend Report. http://www.commtouch.com/downloads/Commtouch_2007_Q4_Email_Threats.pdf, January 2008.

[10] E. Allman, J. Callas, M. Delaney, M. Libbey, J. Fenton, and M. Thomas. RFC4871 - DomainKeys Identified Mail (DKIM) Signatures. http://www.ietf.org/rfc/rfc4871.txt, 2007.

[11] Authentication & Online Trust Alliance. State of Email Authentication and the Internet Trust Ecosystem. http://aotalliance.org/resources/authentication/2008%20AOTA%20Authentication%20Report%2001-30.pdf, February 2008.

[12] P. O. Boykin and V. P. Roychowdhury. *Leveraging Social Networks to Fight Spam*, volume 38(4):61-68. Computer, 2005.

[13] R. Chandhok and G. Wenger. RFC2919 - List-Id: A Structured Field and Namespace for the Identification of Mailing Lists. http://www.ietf.org/rfc/rfc2919.txt, 2001.

[14] M. Delaney. RFC4870 - Domain-Based Email Authentication Using Public Keys Advertised in the DNS (DomainKeys). http://www.ietf.org/rfc/rfc4870.txt, May 2007.

[15] S. Garriss, M. Kaminsky, M. J. Freedman, B. Karp, D. Mazieres, and H. Yu. Re: Reliable Email. In *NSDI*, San Jose, California, May 2006.

[16] J. Golbeck and J. Hendler. Reputation Network Analysis for Email Filtering. In *Conference on Email and Anti-Spam*, Mountain View, California, July 2004.

[17] L. Johansen, M. Rowell, K. Butler, and P. McDaniel. Email Communities of Interest. In *Conference on Email and Anti-Spam*, Mountain View, California, August 2007.

[18] J. Lyon and M. Wong. RFC4406 - Sender ID: Authenticating E-Mail. http://www.ietf.org/rfc/rfc4406.txt, 2006.

[19] J. R. Mastaler. Tagged message delivery agent. http://www.tmda.net.

[20] D. Mazieres. Mail Avenger. http://www.mailavenger.org/.

[21] M. Paganini. ASK: active spam killer. In *USENIX Annual Technical Conference*, pages 12–12, Berkeley, CA, USA, 2003. USENIX Association.

[22] G. Pape. Qconfirm. http://smarden.org/qconfirm/.

[23] Permission Technologies. CleanMyMailbox - Permission-Based Filtering Service. http://www.cleanmymailbox.com.

[24] R. G. Smith, M. N. Holmes, and P. Kaufman. *No. 121: Nigerian advance fee fraud*. Trends & Issues in Crime and Criminal Justice, July 1999.

[25] G. Systems. CertifiedEmail. http://www.goodmailsystems.com/.

[26] B. Taylor. Sender Reputation in a Large Webmail Service. In *Conference on Email and Anti-Spam*, Mountain View, California, July 2006.

[27] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Proceedings of Eurocrypt*, pages 294–311, 2003.

[28] M. Walfish, J. Zamfirescu, H. Balakrishnan, D. Karger, and S. Shenker. Distributed Quota Enforcement for Spam Control. In *3rd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, May 2006.

[29] M. Wong and W. Schlitt. RFC4408 - Sender Policy Framework (SPF) for Authorizing User of Domains in E-Mail, Version 1. http://tools.ietf.org/html/rfc4408, 2006.