STAMP: Toward Reclaiming Email Address Privacy

Kurt Ackermann Camille Gaspard Ramana Kompella Cristina Nita-Rotaru
Department of Computer Science, Purdue University
{keackerm,cgaspard,kompella,crisn}@cs.purdue.edu

I. Introduction

Email has grown into one of the dominant forms of communication in the 21st century. However, email systems were designed without security in mind, thus allowing attackers to abuse the system and send unsolicited email (or spam). The problem of spam has become so severe that recent studies [1], [2] report that over 90% of the emails sent in 2007 were spam, resulting in productivity losses amounting to over $20 billion annually [3]. The negative impact of spam is also amplified by its use in identity theft [4]. Not surprisingly, there has been significant effort during the last few years to develop and deploy solutions to prevent, detect, and filter out spam.

Most current solutions to spam center on content-based filtering (e.g., SpamAssassin [5]), behavioral-based filtering [6], or domain blacklisting approaches, all of which are inaccurate and slow to adapt to the changing face of spam. Methods such as user/domain authentication (e.g., PGP [7], IBE-email [8], and DKIM [9]) and email address obfuscation [10], [11], [12] raise the bar for the attacker but offer only a limited protection against spam. Moreover, these schemes do not provide accountability of email address leakage, which would allow a user to know which untrustworthy parties divulged his address.

We propose STAMP, the *Solicitation Token Authenticated Mail Protocol*, as a server-side solution to filter unsolicited mail from ever reaching the end-user's inbox, as well as allowing the user to revoke inbox access from solicited parties who prove to be untrustworthy with their email access. STAMP employs distributed access control, making use of transitive trust to reduce email solicitation overhead and allow the user's address book to grow organically through trusted entities. We implement a prototype of our scheme as an extensible mail filter plug-in for an industry standard mail server and compare performance against a popular content-based filter.

II. STAMP Protocol Overview

The main goal of STAMP is to provide a mechanism through which a user can control who is allowed to send him email. STAMP ensures that when a user is presented with an email, it originated from a sender who has been explicitly solicited by that user before. We define *solicitation* to be the authorization of a party to send the user mail, and refer to such a solicited party as a *penpal* of the user. The system enables users to revoke inbox access from any penpal who proves to be untrustworthy with his email address (e.g., who divulges it to another party who starts sending spam). In addition, the system allows trusted penpals to grant third parties access to the user's inbox in a secure and accountable fashion. At a high level, our STAMP email protocol is comprised of three main components: *solicitation*, *authentication*, and *revocation*.

Solicitation. A user wishing to receive email from a penpal, explicitly solicits the penpal by establishing a shared secret with him, referred to as a *token*. This token is used in a cryptographic protocol to allow a penpal's email to be authenticated as solicited mail by the user. The requirement of the token ensures that the attacker must not only know the email address, but also to have been authorized by the message recipient through solicitation. Solicitations can optionally be qualified by the user, allowing the user to specify exactly under what conditions he would like to receive mail from the penpal, possibly specifying an expiry time, rate limit, attachment restrictions, etc. To ease the solicitation process, this shared secret token can be accountably extended by a user's penpals to other parties. The distribution of the token is protected from passive adversaries who are eavesdropping.

Authentication. The authentication process occurs at the mail server (MTA) during SMTP transfer, and consists of a HMAC computation over the message contents including the sender and receiver address, with the token as key. The token used to generate the message is queried into the set of active solicitation tokens, and compared against that token's filter rules. If no matching active token is found (i.e. sender and receiver have not performed solicitation to build a token), if a token is found but has been revoked, or if the message does not meet the filtering requirements associated with that token (e.g., too many messages sent already), the message is discarded and never seen by the user. Otherwise, the message is deemed solicited and is placed in the user's inbox.

Revocation. STAMP is designed to operate under the assumption that penpals may not be trustworthy, or may turn treacherous at a later time, divulging email addresses and the corresponding solicitation tokens to third parties. In such cases, our sender revocation mechanism allows a user to identify which trusted penpal disclosed the email address and token, and revoke mail sending privileges from that penpal. When user U receives any item of spam that has passed authentication, it is known that either one of his penpals has begun to send him unwanted mail, or has broken trust and divulged his token to someone who has. In both cases, the response is to revoke the solicitation token used to authenticate the message to prevent any further email using that token from being authenticated. Revocation is a local action that flags the entry corresponding to that particular penpal in the local MTA token table. All subsequent messages in the user's inbox using that token are also discarded, which ensures that for every solicitation token U generates, he will view at most one

piece of spam before access is revoked and the guilty party is identified.

III. Performance Evaluation

We compare the performance of our STAMP prototype mail filter implementation against that of the popular content-based Bayesian-classifier mail filter SpamAssassin [5], using the well-known mail server Sendmail [13]. Each data point is the average of 10,000 message deliveries.

We selected Sendmail because it is the most popular MTA on the Internet [14] and because of its efficient, extensible, industry standard mail filter API, aptly termed *milters* [15]. The milter functionality provides portability of our prototype across versions of Sendmail, as well as other mail servers that support the milter API, including Postfix [16]. In addition, milters provide great control and flexibility in the handling of messages, allowing for the processing of message parts during the course of SMTP transmission.

Our results in Figure 1(a) show that SpamAssassin incurs a classification processing overhead that is more than 3 orders of magnitude greater than our protocol. This processing overhead is directly attributed to the Bayesian classifier in comparison to the less expensive HMAC operations that STAMP performs. STAMP incurs less than 40 ms of processing time for the largest message, while SpamAssassin takes more than 6 seconds to make a classification. We conclude that STAMP incurs linear processing overhead that is insignificant relative to message delivery time, while SpamAssassin incurs an overhead that actually overwhelms the message delivery time.

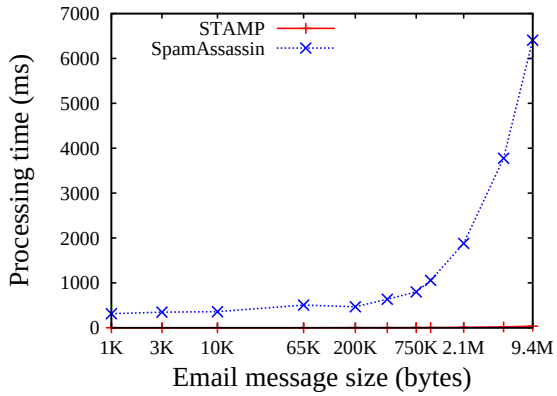
Next we measure the delivery latency as observed from the user perspective. As Figure 1(b) shows, our scheme incurs latency overhead over Sendmail that is imperceptible from experiment noise with message sizes smaller than 2.1MB. SpamAssassin incurs noticeable latency at all message sizes, growing proportionally worse at sizes greater than 750KB. We conclude that SpamAssassin introduces noticeable message delivery latency overhead while STAMP remains nearly identical to the no filtering case.

To summarize, STAMP message authentication is more efficient than Bayesian content classification by orders of magnitude. While our scheme requires an extra step to be performed by the user during *solicitation*, the benefit gained is the perfect classification of messages with accountability for trecherous penpals.

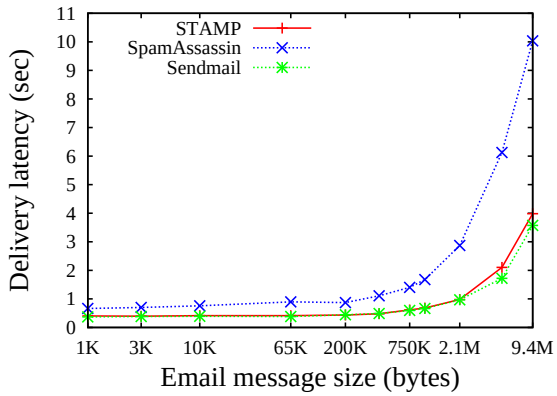
IV. Conclusion

In this work, we proposed a new solution to enable users with efficient and accountable access control to reclaim email address privacy. We motivated our design by arguing that today's anti-spam solutions do not provide a safe mechanism for users to control who can send them email, and as such do not solve the problem of spam in the long-term.

Our solution, STAMP, is based on the efficient use of cryptographic primitives to authenticate solicited senders while enabling users to revoke access once they determine a sender to be untrustworthy. STAMP reduces false-positives when added to current mail servers without compromising performance or user privacy yet leveraging traceability. Current and future work include incorporating a transitive trust scheme to allow trusted penpals to extend the solicitation and revocation schemes.



(a) Server processing time



(b) Message delivery latency

Fig. 1. STAMP micro and macro server performance benchmarks.

References

[1] "Barracuda networks annual spam report." http://www.barracudanetworks.com/ns/news_and_events/index.php?nid=232.
[2] "Commtouch q3 2007 email threats trend report." http://www.commtouch.com/downloads/Commtouch_2007_Q3_Email_Threats.pdf.
[3] "Spam costs billions." http://www.informationweek.com/story/showArticle.jhtml?article
[4] "Gartner: Phishing on the rise in u.s.." http://news.zdnet.com/2100-1009_22-5234155.html.
[5] "Spamassassin." http://spamassassin.apache.org/.
[6] S. V. A. Ramachandran, N. Feamster, "Filtering spam with behavioral blacklisting," in *CCS '07*.
[7] S. Hernan and L. Pesante, "Using pgp to verify digital signatures," *CERT Coordination Center*, 2001.
[8] "Voltage securemail." http://www.voltage.com/products/securemail.htm.
[9] "Domainkeys identified mail (DKIM)." http://www.dkim.org/.
[10] "Safemail." http://safemail.justlikeed.net/.
[11] "Email obfuscator." http://www.killersites.com/webDesignersHandbook/emailObscucato
[12] "Email cloaker." http://emailcloaker.com/.
[13] "Sendmail." http://www.sendmail.org/.
[14] "Security space: Mail (mx) survey." http://www.securityspace.com/s_survey/data/man.200711/mxsurvey.html.
[15] "Milters." https://www.milter.org/.
[16] "The postfix project." http://www.postfix.org/.