

A Policy Driven Approach to Email Services *

Saket Kaushik
ISE Department, MSN 4A4
George Mason University
Fairfax, VA 22030, U.S.A.
skaushik@gmu.edu

Paul Ammann
ISE Department, MSN 4A4
Center for Secure Inf. Sys.
George Mason University
Fairfax, VA 22030, U.S.A.
pammann@gmu.edu

Duminda Wijesekera
ISE Department, MSN 4A4
Center for Secure Inf. Sys.
George Mason University
Fairfax, VA 22030, U.S.A.
dwijesek@gmu.edu

William Winsborough
ISE Department, MSN 4A4
Center for Secure Inf. Sys.
George Mason University
Fairfax, VA 22030, U.S.A.
wwinsbor@gmu.edu

Ronald Ritchey
Booz Allen & Hamilton
Falls Church, VA, U.S.A.
ritchey_ronald@bah.com

Abstract

The primary original design goal for email was to provide best-effort message delivery. Unfortunately, as the ever increasing uproar over SPAM demonstrates, the existing email infrastructure is no longer well suited to the worldwide set of email users - particularly email receivers. Rather than propose yet another band-aid solution to SPAM, this paper rethinks email from the requirements perspective, albeit with the constraint of designing a system suitable for incremental adoption in the current environment. Our result to this exercise is a policy driven email service in which the interests of each principal can be articulated and accommodated. Our scheme rewards faithful senders with better quality of service and discourages misbehavior. Our scheme provides receivers with policy-driven control over whether and how a given message appears in the recipients mailbox.

1. Introduction

We replace the current notion that an arbitrary user has an implicit right to place a message in anyone's mailbox with a scalable, attribute-based access control policy in which a sender's access to a recipient's mailbox is controlled by the recipient. In this work we view a mailbox as a network resource of the recipient and email messages

as access requests from senders, requesting write access to a mailbox. In addition to the sender and receiver, there are two other principals directly associated with the email pipe, namely the sender's and receiver's email service providers. Also, there are important third party participants in the scheme that provide reputation and adjudication services. We take the position that any solution that does not address the needs of all of the principals is unlikely to achieve its objectives. Our approach complements many existing proposals to improve email, such as the Project Lumos's initiative to control bulk email servers and Razor's reputation service for identifying SPAM.

The place to start with email service is sender authentication. Any approach to email that fails to incorporate some notion of sender authentication seems limited, at best. Somewhat surprisingly, sender authentication is widely believed to be a hard problem, usually due to an assumption that a widespread PKI infrastructure is necessary to support it. For example, Eric Allman's excellent overview of the SPAM problem [1], [2] takes exactly this position. However, as noted in the Project Lumos's white paper [6], most senders send email through an account with an Email Service Provider (ESP), and there is typically some sort of login required before mail can be sent. As for the ESP itself, there is already some infrastructure in place that allows mail servers to identify themselves when transmitting email. The fact that there are vastly fewer mail servers than there are email senders makes the task of securely authenticating ESPs (or, more specifically, an ESP's Mail Transfer Agent or MTA) far more manageable.

*This work supported in part by the National Science Foundation under grants: CCR-TC- 0208848, CCR-0113515 and CCR-0325951.

At the point at which a message originates, the sender's ESP typically can associate a password-authenticated account with the message, or, stated another way, the ESP knows which of the ESP's customers sent the mail. While this is not true for all email, it is true (or could be made to be true with relatively little effort) for most legitimate email. There are valid exceptions, such as deliberately anonymous email. While anonymous email is outside the scope of the current work, our policy-based approach certainly does not prohibit anonymous email. Rather, it is simply up to the recipient's policy whether and how to handle such email. In some cases, such as a corporate organization, the ESP knows quite a bit about the customer. Even in cases such as a HotMail message, the ESP is in a position to know, for example, how many email messages that customer has sent in the past hour. If the ESP passes this knowledge to the receiver, the receiver will be in a better position to decide how to handle the mail.

From our perspective, we view the level of sender authentication provided by email accounts at ESPs as relatively weak, but serviceable. The fact that the approach is essentially implemented and deployed makes it far more attractive than a full blown PKI approach. One of the interesting attributes of email account approach is that it gives the sender's ESP a central role in a policy based email approach. Specifically, an ESP is in a position to know what volume of email the sender has transmitted in the past, what complaints, if any, have been lodged against sender, and so forth. If information about how an ESP manages its customers is made public, which seems to be the direction in which the internet community is traveling, then ESPs that permit senders to misbehave will suffer a backlash from the outside world, which will, in turn, drive legitimate senders away from the ESP. Thus the sender and the sender's ESP each can have legitimate, but widely different, interests in the transmission of a given message. Sensible email policies need to accommodate both of these parties.

On the receiving end, users typically receive email through another ESP. In correspondence with the sender, both the receiver and the receiver's ESP have legitimate, but different, policy interests in how to process incoming email. The fact that there are also privacy interests involved complicates the issue. For example, a receiver may not wish to make public the fact that he or she rejects (or accepts) certain types of email.

Figure 1 illustrates the principals in a secure email pipe. In addition to the sender, the receiver, and the corresponding ESPs, the figure shows third party reputation and adjudication servers. These services provide evaluations of particular messages, senders, or ESPs. Recipients, or recipient ESPs provide the raw data that these servers use to make their evaluations. Adjudication services provide a third party mechanism for satisfying obligations that may

be incurred as the result of delivering a mail. For example, a sender or a sender's ESP may post a small bond as a guarantee to ensure recipient satisfaction.

The key contributions of this paper are the content of sections 3 and 4. Specifically, in section 3, the heart of the paper, we contribute 3 new types of policies. The first type of policy, a Service Level Agreement Policy (SLAP), addresses how a receiving ESP decides to interact with a given sending ESP that has announced that it has mail. The second type of policy, a Message Scheduling Policy (MSP), is the output of a SLAP evaluation, and specifies how each message at the sending ESP should be treated. The third type of policy, a Message Resource Allocation Policy (MRAP), encodes the specific requirements of an individual mail recipient, and is used to determine how (and whether) messages are presented to the actual human recipient. In section 4, we explain how the SMTP protocol can be extended to accommodate each of these three policies, thereby making an argument that the policies are plausible given the current protocol for sending email.

The paper is organized as follows. In the next section, we review the current mechanism for exchanging mail, the SMTP protocol, and enumerate its shortcomings. In sections 3 and 4, we present our main contributions in the form of 3 new types of policies and their relation to the existing SMTP protocol and introduce a candidate base set of predicates that principals in an email exchange could use to evaluate a given email message. Section 5 discusses related work, and section 6 concludes the paper.

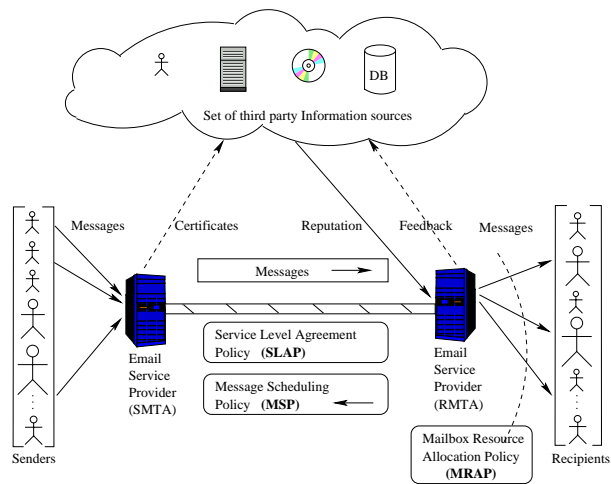


Figure 1. Principals in a Secure Email Pipe

2. Current SMTP Protocol

The current Internet mail system (or e-mail system) has three major components: *user agents*, *mail servers*, and the

Simple Mail Transfer Protocol (SMTP) [21]. A typical message starts its journey from the sender's user agent, travels to the sender's mail server, and then travels to the recipient's mail server, where it is deposited in the recipient's mail box [12]. SMTP, defined in RFC 2821 [21] transfers messages from the sender's mail server to the recipient's mail server. First, the SMTP client (running on the sending mail server host) has TCP establish a connection on port 25 to the SMTP server (running on the receiving mail server host). If the server is down, the client tries again later. Once this connection is established, the server and client perform some application-layer handshaking. During this handshaking phase, the SMTP client indicates the email address of the sender and the email address of the recipient. Next, the client sends the message. The client then repeats this process over the same TCP connection if it has other messages to send to the server, otherwise it instructs TCP to close the connection [12]. An simple explanation (not complete) of the protocol is provided in figure 2.

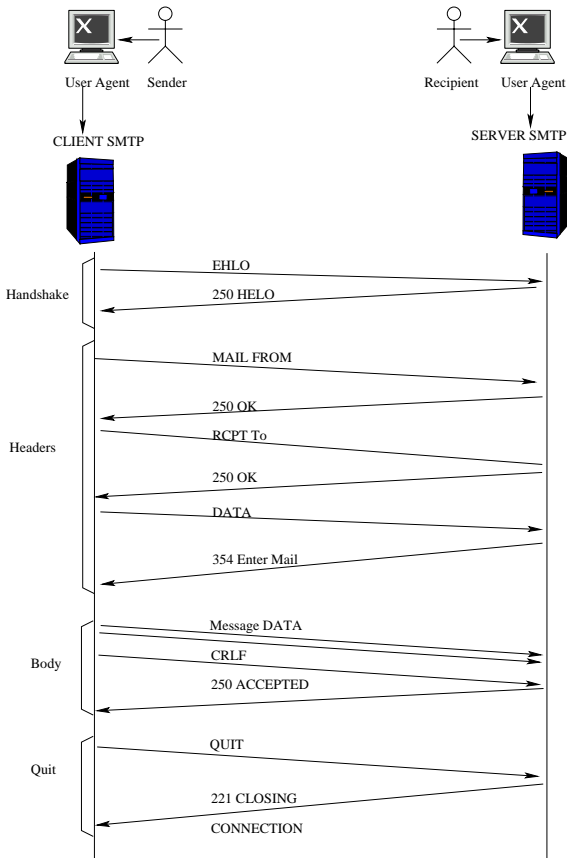


Figure 2. Current SMTP Protocol

Notation 1

Internet mail system entities discussed here are referred

to by their descriptions in [12]. For ease of understanding and brevity following notations are used interchangeably:

Symbol	Description
Sender, Recipient	Human users
MTA	Mail Transfer Agent [21]
ESP	Email Service Provider

Table 1. Industry Standard Symbols

Symbol	Description
SMTA, RMTA	Sending MTA, Recipient MTA
SESP, RESP	Sending ESP, Recipient ESP
SLAP	Service Level Agreement Policy
MSP	Message Scheduling Policy
MRAP	Mailbox Resource Allocation Policy

Table 2. Symbols introduced in the paper

Pseudo code in figure 13 in the Appendix provides a brief description of the algorithm that manages the state changes at the SMTA. A similar algorithm is presented in figure 12 in the Appendix which shows state changes for the RMTA.

2.1. Problems with the current protocol

The essence of the problem with original protocol is that the RMTA acts as a slave of the SMTA. The RMTA has no discretion regarding which and how many messages it receives, nor from whom. Numerous times in the past, this fact has been exploited by malicious entities to send unsolicited mail or spam [17], to the detriment of powerless recipient's. RFC 3207 [23] and RFC 2554 [22] add to SMTP support for authentication of SMTAs. They prevent impersonation of SMTAs, and so enable the RMTA to refuse service to specific SMTAs. This is an improvement, but does not provide a basis for accepting some messages (e.g., a personal message) from a given SMTA, but not others (e.g., bulk). It gives no basis for assessing whether a given, clearly identified SMTA is trustworthy. Moreover, it provides no control to the recipient, who is clearly an interested party. Factors other than the origin of a message could have an impact on its treatment, such as how the message is characterized by its sender (personal, bulk, ...), the current work-load level at the RMTA, or whether there are active virus alerts or the RMTA is under denial-of-service attack. In essence, what is absent in current email protocol is a framework for managing the allocation of mail-delivery resources based on the myriad of factors that could be of concern to the interested parties, i.e. the owners of those resources. In the next section we propose such a framework.

3. Policy driven email transfer

In this section we present a policy architecture for controlling the email pipeline. The goal of this architecture is to give the interested parties (*viz.*, the RESP and the recipient) control over the use of email-delivery resources (*viz.*, delivery service and mailbox access), thereby overcoming the shortcomings of the current SMTP protocol discussed above and enabling RESPs and recipients to prevent what they consider to be misuse of their resources.

The architecture consists of three forms of policy, introduced in each of the three subsections below. The first is associated with the RESP, and specifies what SESP's it will accept messages from, when, and under what terms; we call this *Service Level Agreement Policy (SLAP)*. The second is optionally provided by the RESP(RMTA) to the SESP(SMTA) to specify which message should be sent when; we call this *Message Scheduling Policy (MSP)*. The third is specified by the recipient, and is used to provide customization concerning whether email is accepted, and to which mailbox folder; we call it *Mailbox Resource Allocation Policy (MRAP)*.

3.1. Service Level Agreement Policy (SLAP)

Connection-level actions of an RESP(RMTA) are governed by its *Service Level Agreement Policy (SLAP)*, which is pictured in figure 3.

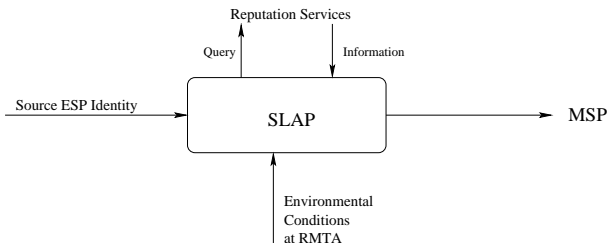


Figure 3. Service Level Agreement Policy (SLAP)

The inputs to SLAP include identity of the SESP, environmental conditions at the RESP, and the reputation of the SESP. This reputation is presumed to summarize past experiences in dealing with the SESP. These may be experiences that the RESP has had, or experiences had by third parties, which are then maintained and distributed by reputation services. The latter could take the form of white lists or black lists (such as provided by Open Relay Blacklists), or they could take some other, more dynamic form. The essential requirement is that the reputation service characterizes the past behavior of SESP's so that the SLAP can

give preferential treatment to SESP's with long records of favorable behavior. Relevant environmental conditions might include work load at the RMTA or Virus Alerts, etc. Together these inputs characterize the current SMTP session; based on them, the SLAP defines (*i.e.*, returns) a specification of what to do with messages waiting for transmission. Candidate specifications include: send them now (“as is”); send them later (“later”); do not accept messages from this SESP (“never”). In the most general case, however, this specification is itself a policy, called a *Message Scheduling Policy (MSP)*, that considers each message in turn, and determines whether and when it should be transmitted. MSP will be discussed further in the next section. SLAP's are not shared, so their format is organization-determined.

Example 1 A certain ESP uses SESP identity and reputation to map each would-be sender to one of the following profiles: Business Partner, Trusted, Suspect, Black-listed, Unknown. Trusted SESP's are trusted by the ESP (RESP) to receive an MSP, and to apply it to waiting messages to determine whether and when to transmit them. The policy format used by the ESP is as follows:

```

if(profile==Partner) return "as is"
if(profile==Black-listed) return "never"
if(load==heavy && profile==Trusted) return MSPheavy
if(load==light && profile==Trusted) return MSPlight
if(Virus alert && profile==Suspect) return "later"
if(load==heavy && profile==Unknown) return "later"
:

```

MSP_{heavy} indicates that messages labeled “bulk” should be delayed, while *MSP_{light}* permits them to be transmitted.

3.2. Message Scheduling Policy (MSP)

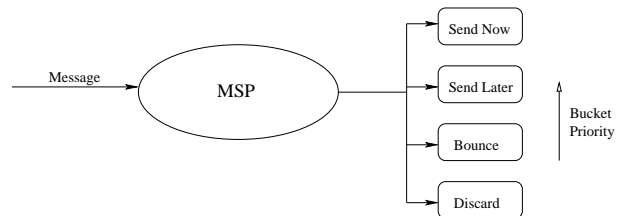


Figure 4. Message Scheduling Policy (MSP)

Message Scheduling Policy (MSP) is applied to each message awaiting transmission to determine whether and when it should be sent from the SMTA to the RMTA. The scheduling of a message will depend on characteristics of the message itself and of its sender, as well as the means of sender authentication. We model each of these characteristics as a predicate that takes a message and returns true if the

message, its sender, or the SMTA involved has the characteristic in question; otherwise it returns false. The following is a (partial) list of such predicates:

Sender Attributes

1. Sender Auth: { Certificate | Password | None }
2. Reputation of sender provided by sending ESP: { Reputation Service specific }
3. Sender Activity in time \mathcal{T} : { No. of Messages in \mathcal{T} }
4. Sender membership duration: { Duration of association with ESP }

Message Attributes

1. Bonded: { Guarantor, Bond }
2. Virus Scanned: { Scanner, version, patch-level }
3. Project Lumos attribute: { PLT-Personal | PLT-Service | PLT-First Class | PLT-Bulk }

MSPs are of two types: *portable* and *local*. The difference lies in where the policy is evaluated. A portable MSP is transmitted from the RMTA to the SMTA, which is trusted to apply the MSP to determine which messages to transmit. A local MSP can be used when the RMTA wishes to filter the messages transmitted, but does not wish to entrust the MSP and its application to the SMTA. When local MSP is used, the SMTA must transmit messages headers, which the RMTA then uses to evaluate the MSP on each waiting message. RMTA then provides the SMTA with instructions regarding if and when to transmit the waiting messages.

The format of local MSP is ESP-determined, but, because it is evaluated remotely, the format of portable MSP must be universal. In either case, the RMTA has the capability to check whether its MSP was adhered to by the SMTA, and contribute this information to build SESP reputation.

Example 2 There are four messages $m_1 \dots m_4$ in the transmit queue at the SMTA. These messages have following characteristics:

$m_1 \rightarrow$ sender auth: Password, Lumos-labeled Bulk

$m_2 \rightarrow$ No Sender authentication

$m_3 \rightarrow$ Bonded Mail, No sender authentication

$m_4 \rightarrow$ sender auth: Password, Lumos-labeled Personal

Given its current load and environment, the MSP of the RMTA for the Sender in question is:

if(Password auth && Lumos-labeled Personal) send now

if(Bonded) send now

if(Lumos-labeled Bulk) delay

if(No sender authentication) discard

The application of this MSP to message (headers) in the Sender's transmit queue yields the following:

Send now $\rightarrow m_4, m_3$

Delay $\rightarrow m_1$

Discard $\rightarrow m_2$

3.3. Mailbox Resource Allocation Policy (MRAP)

Mailbox Resource Allocation Policy (MRAP) is specified by mail recipients and controls the utilization of their mailbox folders. User mail-reception preferences are captured and converted to MRAPs, which are evaluated at the user's ESP as part of the process of delivering a message to the user. MRAPs are pictured in figure 5. The input to MRAP includes: the *complete* message (including content); sender identity, characteristics, and reputation; SESP identities and reputations; message attributes; mailbox state. The result of MRAP evaluation is either to discard or bounce the message, in which cases the message is not delivered to the mailbox, or to assign a discrete category (*i.e.*, a mailbox folder) to route the message to.

Example 3 The five messages shown in table 3 are sent. Evaluating the recipients' MRAPs on those messages has the following results:

Alice evaluates m_1 by using reputation service X \rightarrow discards.

Bob evaluates m_2 by using reputation service X \rightarrow accepts — Bob's preferences lead him to a different action than Alice's, although he is sent the same message and gets the same information about it (from X).

Carol gets the same message as Alice and Bob, but does not consult a reputation service. Instead, she notes that the sender of m_3 has sender activity > 100 messages / hour \rightarrow discards.

Bob evaluates m_4 with reputation service Y, which ranks it 'unobjectionable'; Bob then notes that his Advertisement folder is over 80% full \rightarrow rejects.

Bob evaluates m_5 with reputation service Y, which ranks it 'interesting'; notwithstanding that his Advertisement folder is over 80% full \rightarrow accepts.

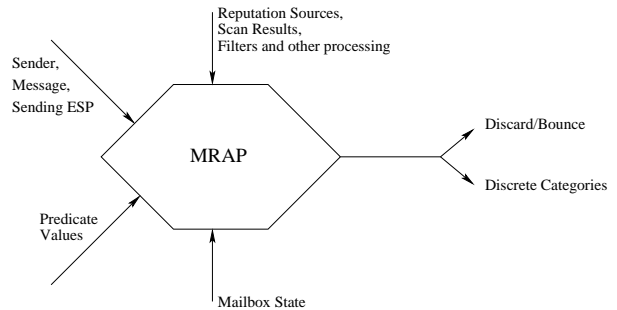


Figure 5. Mailbox Resource Allocation Policy (MRAP)

Message	From	To	Content
m_1	unknown	Alice	arbitrary string
m_2	unknown	Bob	same as m_1
m_3	unknown	Carol	same as m_1
m_4	commercial	Bob	advertisement
m_5	commercial	Bob	advertisement

Table 3. Messages with some characteristics

4. Proposed Extensions to SMTP

In this section we present how to modify the SMTP protocol to implement the decisions made by policy architecture introduced in section 3. The revised protocol is described in figure 6 and three additional message headers are described in figure 7.

To indicate policy based decision support, the EHLO [24] command is replaced by SHLO. SHLO should support all features of EHLO and in addition indicate extended handshake as shown in figure 6. SHLO implies authentication of MTA (as in [22],[23], [14] etc.), and in addition the SMTA would sign a hash of all the message headers and the hash of complete message and include it in the message as additional headers (see figure 7).

4.1. Extended Handshake

The first extended handshake *command* SHLO includes keywords for : server’s identity, Message characterizations and other attributes like reputation references or certifications (keywords not defined) apart from the STARTTLS [23] or AUTH [22] and other usual EHLO parameters. The RMTA completes the negotiation by returning the keywords it supports. Figure 6 skips rest of the handshake commands and replies but details the identity and parameters transferred from SMTA to RMTA and the RMTA response. The SMTA reports the subset of the base predicates it supports described in section 3.2 (in the X-attribute header -figure 7) for the RMTA to define the MSP. Reputation references and certifications are included to get a higher priority SMTP session with the RMTA.

Reputation information is important for decision making at the RMTA. Source of reputations could be past behavior [8] with the same SMTA (maintained by RMTA) or a distributed information sharing mechanism like SpamNet [4] etc., open relay blacklists or provided by the SMTA (certifications on the lines of TRUSTe [26]). The onus to collect reputation lies on the RMTA (line 5 fig 8). RMTA evaluates these parameters based on its *Service Level Agreement Policy* and decides on the *Message Scheduling Policy* to apply to the Session. As will be discussed further in the next

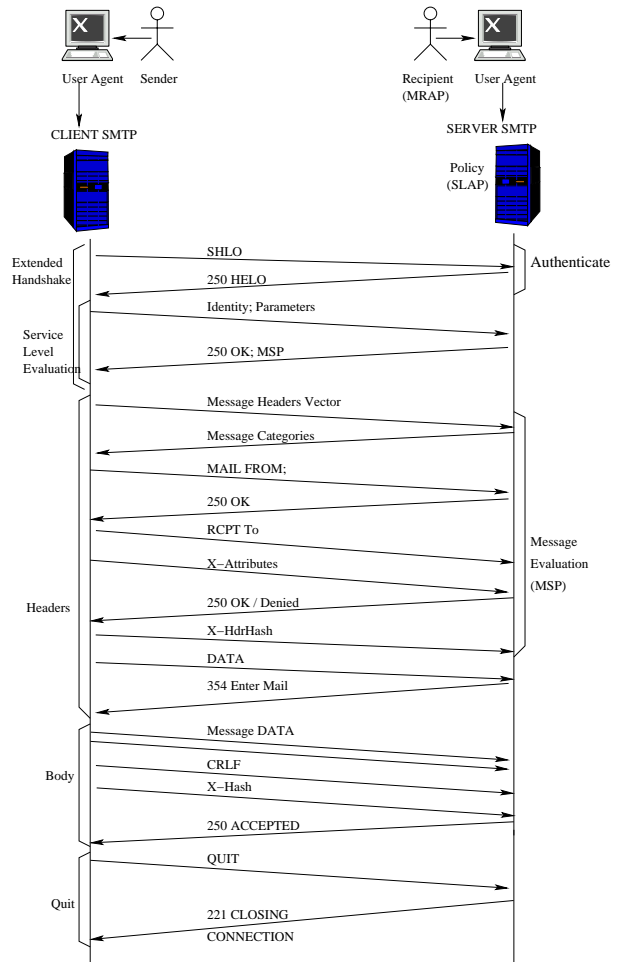


Figure 6. Proposed extensions to SMTP Protocol

Usual Headers [18]

- Mail From: Sender address

⋮

Additional Headers for policy support

- X-Attributes: Comma separated attribute list
- X-HdrHash: Signed hash of headers
- X-Hash: Signed hash of the complete message

Figure 7. Message Headers for Policy Support

```

1  if(state == connected)
2    then Wait for request;
3    if(request.hasParams)
4      then Parse params;
5    Reputation = getRep(params.ID);
6    MSP = SLAP(params, Reputation, Environ);
7    if (MSP.type==portable)
8      then Response = {Code, MSP};
9     state=portable;
10   else Response = {Code, send-Headers};
11     state=local;
12   Send Response;

```

Figure 8. Extended Handshake at RMTA

```

1  Send Command = {Identity, Params};
2  Wait for response;
3  if(response == 250 OK && response.hasParams)
4    then Parse Params;
5    MSP = response.Params;
6  if (MSP ≠ send-Headers )
7    then state = schedule messages;
8  else state= send Vector;

```

Figure 9. Extended Handshake at SMTA

section, message scheduling can be performed at either the SMTA or at the RMTA. In the latter case, the RMTA sends in the MSP parameter a keyword meaning “send all headers,” which indicates to the SMTA that it should send a Vector of all transmit queue messages to the RMTA before any message is transferred. The pseudo code is described in lines 3 to 12 figure 8 (which replaces the steps between lines 5 and 6 in figure 12). Thus the handshake is completed.

4.2. Message Evaluation

At the SMTA, if the MSP parameter of the 250 *OK* message is the keyword for “send all headers”, then the next step for the SMTA is to provide a vector of all message headers in the transmit queue. It then waits for RMTA’s response on the message scheduling of its messages (see lines 3 to 8 in figure 9). Otherwise, the SMTA proceeds to apply the received MSP to outbound queue and send messages in *send now* queue beginning with message headers (figure 10).

On receiving a message headers vector, the RMTA looks up its MSP to sort messages into different buckets of *send now*, *later*, *discard* etc It then sends the updated vector back to the SMTA (figure 11).

In the case that the MSP specified by the RMTA is portable, both parties proceed to the message-transfer stage.

```

1  if(state == send Vector)
2    then (command = Header Vector(message queue))
3    Send command;
4    Wait for response;
5    if (response.hasParameters) then
6      MSP = response.Parameters;
7      state = schedule messages;
8  if(state == schedule messages)
9    Send now queue = response.SendNow;
10   Send later = response.SendLater;
11   Discard = response.Discard;
12   state = send sender; ;

```

Figure 10. MSP set up at SMTA

```

1  if(state==local) && request.hasParameters)
2    then (Header Vector = request.Parameters)
3     Send now queue = getNow(Header Vector);
4     Send later = getLater(Header Vector);
5     Discard = getDiscard(Header Vector);
6     response = Order(Send now queue, Send later, Discard);
7     Send response;
8     state = connected;

```

Figure 11. MSP set up at RMTA

On receipt of headers for a message, the RMTA can optionally verify that the MSP has been applied and, after the X-HdrHash header is transmitted, it can send back 250 OK response or a Denied response (see figure 11).

After a message arrives at the RMTA, the current SMTP session is not concerned with its further handling. Its disposition thereafter is controlled by MRAP, which reflects user preferences, and can depend on the complete body of the message. MRAP-determined bounce decisions are relayed as error messages out of band from the session in which the original message was transferred. The content of such feedback and whether any is provided depend on the level of trust that the RMTA places in the SMTA. This is a matter of policy for the RMTA, which can be controlled via an additional component of the SLAP policy.

5. Related Work

At the forefront of solutions against fraudulent email or spam are *laws and legislative provisions*. Laws are essential and helpful but certainly not the silver bullet we have been searching for, as summed up in [2]. They have limited jurisdiction and are ineffective against global spammers. Laws without technical support can be ill-directed. We aim to

provide such a basis for laws to be effective.

Content based filtering solutions parse the content of a message (for keywords or phrases) to determine the probability of it being spam. Some of these approaches are showcased in ([9], [20], [28]). Naive Bayesian filters are considered most effective of all filtering solutions. Even though false positives might not be a big problem with them [11], the approach is reactive and does not discourage spammers from spamming. In fact, they work to defeat such filters and simply increase the number of messages sent. Our approach can take advantage of such pre-delivery processing of messages and add other essential mechanisms to be more proactive in reducing spam messages transmitted.

Project Lumos [6] includes the same notion of two-level sender authentication that we propose to use. The difference with the current work is that Lumos is set up, at heart, to serve the interests of the large commercial bulk mailers. The idea is to register bulk mailers, require them to respond to reports of abuse, and publish the corresponding reputations. If Lumos really does make headway in the marketplace, our scheme would be easier to implement, since we rely on the similar notions of sender authentication. Also, Lumos would be an excellent reputation service to include in an MSP or MRAP. Other examples of proposed or existing *reputation services* are Razor [19] and CloudMark [4], which aim to identify individual SPAM messages, and SmartScreen [16], which is a Microsoft effort with the same objective. Both Razor and SmartScreen would fit well with the approach that we propose. A weaker source authentication scheme proposed by Boykin and Roychowdhury [3] takes advantage of social networks to construct a list of trusted senders. The authors achieve limited success with this approach and note that the technique alone is insufficient to catch all spam in email. Spammer monitoring a network can get by such *whitelist* based solutions. Strong authentication based solutions [25] are clearly unscalable for email applications.

The third genre of email abuse solutions involve *economic solutions* like bonded mail [13], e-stamps [5], Penny Black [15] or computational or memory based costs [10]. These solutions would reduce spam drastically, but every one has to bear the cost and it is against the open internet flavor. In general, we would like the email service not to punish legitimate users for misbehavior of others. However, some organizations might have a realistic need for such a solution and we provide a model in which economic solutions can work on an incentive, rather than a punitive, basis. Specifically, it is straightforward to write an MSP or MRAP so that mail that otherwise might be reject is accepted if it is bonded. A sender who really wants to get through, and is prepared to pay a price set by the receiver, will get through. The important aspect we introduce is that the receiver, through an MRAP, has control over the ultimate de-

cision of whether to accept a mail message with a specified set of attributes. Through appropriate additional attributes, buying interrupt rights ([7],[13]) is possible in our scheme.

Challenge/Response systems [27] hold promise by requiring the sender to pass challenge-response tests for human initiation; which would be a big improvement, although, overlaying SMTP with an interactive protocol signifies a substantial change in the way email is delivered and must be mandated by careful network traffic studies. Also, distinguishing a human from an automated program may not be enough for spam control, as recipients may need a finer grained control over the messages they accept or reject. In any case, our approach would only benefit from a challenge response mechanism as it can be included as another attribute to specify rules around.

Our approach provides an umbrella under which most realistic needs can be expressed as policies and negotiated during the SMTP connection set up. This involves extending the SMTP, but adoption is possible incrementally without breaking existing implementations.

6. Conclusion

In this paper, we have argued that email needs an end-to-end overhaul if it is to continue to serve as the valuable resource that it has been in the past. In particular, small, point patches cannot correct the fundamental design problem of the existing email system, namely, that by default arbitrary users have write access to arbitrary mailboxes and filtering solutions though effective are only a stop gap solution. As noted by author of `sendmail`, solutions which do not increase the cost of sending spam are insufficient as the spammer can get around such a solution. In essence, we need a feedback mechanism to allow recipients select the clients they communicate with and hence discourage the process of sending mails at will. We began our review by noting that a weak but serviceable notion of sender authentication for email is within reach of the existing internet infrastructure. Our main contributions are three types of policies:

- The first, Service Level Agreement Policy or SLAP, allows a RESP to determine the conditions under which it will communicate with a given SESP ((SLAP) in figure 1). Past experience with the sending ESP, third party evaluations of the SESP's reputation, the set of services supported by the SESP, and the RESP's load level all serve as inputs into a SLAP (figure 3).
- The result of evaluating SLAP is a Message Scheduling Policy or MSP, which instructs the SESP on whether and when to send each particular message ((MSP) in figure 1). In contrast to the existing SMTP protocol, which sends every message as soon as possible in first come, first served order, an MSP evaluates the specific attributes of each email message and

uses these attributes to assign the message a priority for transmission, a deferred transmission time, or a flat out rejection (figure 4).

- The last, Message Resource Allocation Policy or MRAP, allows an individual user to specify how incoming mail should be evaluated prior to appearing in the (human) user's mailbox. The attributes used by MSP are all available to the MRAP as well, and it is expected that the MRAP will use these attributes in a more restrictive manner than does the MSP ((MRAP) in figure 1). In addition, an MRAP may request third party reputation evaluation, and also base decisions on local state information, such as how full the user's mailbox is. The net result is to give each receiver fine-grained, attribute-based control over their mailbox.

We show how to integrate each of SLAP, MSP, and MRAP with the existing SMTP protocol, thereby completing our case that our proposal is reasonably compatible with existing email infrastructure. Twin issues of controllability achieved through these policies, and resolution of policy conflicts are not discussed in this work. Intuitively, the policies introduced allow a rich set of rules to be expressed (like selectiveness in SMTP communications based on SESP identity and prioritizing 'good quality' messages) and in future work we propose to investigate if these are enough to address the spam problem.

We envision an incremental adoption scheme in which SESPs that adopt the scheme are rewarded with better service for their customers. When adoption reaches a given point, those who lag behind in adoption will find themselves discriminated against by the SLAP policies of RESPs.

We deliberately do not attempt to detect misbehavior in our scheme. Instead, we expect that misbehavior will be noted externally - in part by end users who file complaints. Mechanisms to support this are already quite common on the Internet. The hashes on mail provided by the sending ESP help regulate complaints by ensuring that a actual email message can verifiably be associated with every complaint. Feedback about misbehaving ESPs or end users makes its way into the application of SLAP, MSP and MRAP policies to future mail.

References

- [1] E. Allman. The economics of spam. <http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=108>.
- [2] E. Allman. Spam, spam, spam, spam, spam, the ftc and spam. <http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=66>.
- [3] P. O. Boykin and V. Roychowdhury. Personal Email Networks: An Effective Anti-Spam Tool, February 2004.
- [4] CloudMark's SpamNet. <http://www.cloudmark.com>.

- [5] E-Stamps. <http://www.templetons.com/brad/spam/estamps.html>.
- [6] Email Service Provider Coalition. Project Lumos. http://www.networkadvertising.org/esp/lumos_white_paper.asp, Sept 2003.
- [7] S. Fahlman. Selling interrupt rights: a way to control unwanted e-mail and telephone calls. *IBM Systems Journal*, 41(4):759-766, November 2002.
- [8] E. Friedman and P. Resnick. The social cost of cheap pseudonyms. In *Journal of Economics and Management Strategy*, Aug 1999.
- [9] P. Graham. A plan for spam. In *2003 Cambridge Spam Conference Proceedings*, 2003.
- [10] Internet Mail 2000. <http://cr.yp.to/im2000.html>.
- [11] Kristian Eide. Winning the War on spam: Comparison of Bayesian spam filters. <http://home.dataparty.no/kristian/reviews/bayesian/>.
- [12] J. F. Kurose and K. W. Ross. *Computer Networking: A top down approach featuring the internet*. Addison Wesley, second edition, 2003.
- [13] T. Loder, M. V. Alstyne, and R. Wash. Information asymmetry and thwarting spam. Technical report, University of Michigan, 2004.
- [14] Microsoft Corporation and Sendmail Corporation. Email Caller-ID in <http://www.internet-magazine.com/news/view.asp?id=3930>.
- [15] Microsoft Corporation's Penny Black Project. <http://research.microsoft.com/research/sv/PennyBlack/>.
- [16] Microsoft Corporation's SmartScreen Press Release. <http://www.microsoft.com/presspass/press/2003/nov03/11-17ComdexAntiSpamPR.asp>.
- [17] S. Petry. Port 25: The gaping hole in the firewall. In *Proceedings of ACSAC'02 Annual Computer Security Applications Conference*, Dec 2002.
- [18] M. Pop. Comparative study of electronic mail systems.
- [19] V. V. Prakash. <http://razor.sourceforge.net/>, March 2000.
- [20] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.
- [21] Simple Mail Transfer Protocol. RFC 2821, Apr 2001.
- [22] SMTP Service Extension for Authentication. RFC 2554, March 1999.
- [23] SMTP Service Extension for Secure SMTP over Transport Layer Security. RFC 3207, Feb 2002.
- [24] SMTP Service Extensions. RFC 1869, Nov 1995.
- [25] T. Tomkins and D. Handley. Giving email back to the users: using digital signatures to solve the spam problem. In *First Monday* 8, 9 in http://firstmonday.org/issues/issue8_9/tomkins/index.html, September 2003.
- [26] TRUSTe. <http://www.truste.org>.
- [27] Visnetic MailPermit. <http://www.deerfield.com/products/visnetic-mailpermit/requirements/>.
- [28] W. S. Yerazunis. Sparse binary polynomial hashing and the CRM114 discriminator. In *2003 Cambridge Spam Conference Proceedings*, 2003.

Appendix

```

1 state = not connected;
2 while (1) (
3   Wait for request from client; /* initiation */
4   if (request=HELO AND state == not connected)
5     then state = connected;
6     Send (250 OK) response;
7     Wait for request;
8   else state = not connected;
9   if (request=MAIL TO && state == connected)
10    then state = received sender;
11    Send (250 OK) response;
12    Wait for request;
13    if (request = RCPT TO && state == received
14       sender)
15      then state = received rept;
16      Send (250 OK) response;
17      Wait for request;
18    if (request = DATA && state == received rept)
19      then state = receiving data;
20      Send (334 Enter Mail) response;
21      Wait for request;
22    if (request = CRLF && state == receiving data)
23      then state = received data;
24      Send (250 OK) response;
25      Wait for request;
26    if (request = QUIT && state == received data)
27      then state = not connected;
28      Send (221 Connection closing) response;
29 )endwhile

```

Figure 12. Parts of RMTA algorithm for communications with SMTA

```

1 while (1) (
2   Wait to receive message from clients; /* Message
3   Queue initialization. (Network service listening on
4   port 25)*/
5   state = not connected;
6   if (message received) then append to message
7   queue;
8   while (message queue not empty)(
9     message = message queue.next();
10    if (state == not connected)
11      then state = connecting;
12      Send HELO / EHLO to SMTP server;
13      Wait for response;
14      if (response == (250 OK))
15        then state = send sender;
16        else break;
17      if (state == send sender)
18        then Send MAIL FROM: sender@domain.com;
19        Wait for response;
20        if (response = (250 OK))
21          then state = send recipient;
22          else remain in line 15;
23        if (state = send recipient)
24          then Send RCPT TO: recipient@recipient.com;
25          Wait for response;
26          if (response = (250 OK))
27            then Send DATA;
28            else remain in line 21;
29          Wait for response;
30          if (response = (354))
31            then input message body;
32            state = data;
33            else remain in line 25;
34          if (message body.end()) then Send CRLF;
35          Wait for response;
36          if (response = (250 OK))
37            then go to line 5;
38            else append message in message queue;
39          )endwhile
40 )endwhile

```

Figure 13. Parts of SMTA algorithm for communications with RMTA