INTERNET-DRAFT
Expires: February 9, 1999
Filename: draft-coulter-pmap-00.txt

D. Coulter
August 9, 1998

Proxy Mail Address Protocol

Status of this Memo

Abstract

This specification defines a service that manages a type of disposable Internet mail address known as a proxy address. Proxy addresses conform to the standard addressing scheme [RFC 822] and exist in the same address space, but act as aliases for regular, fixed addresses. Users may own many at a time and allocate and deallocate them at will.

Proxy addresses offer a defense against junk mail by allowing individuals to control access to their mailboxes by creating addresses for specific contacts or purposes. If unwanted mail arrives addressed to a proxy, the user may delete or suspend the proxy address to remove the intrusive sender's means of accessing the mailbox.

The service also defines a distinct command-response interface for use between client and server implementations to conduct management chores.

Table of Contents

1. Introduction

Perhaps the most intrusive form of Internet mail abuse, the unsolicited mail problem is a chronic aggravation that users are generally powerless to stop or prevent. Many forums have played host to endless criticism of the problem, particularly leveled against those who engage in the activity and their tactics, but few solutions have come to the forefront that provide users an adequate defense. While economic and social influences define the human side of the problem, the permissiveness of the Internet mail architecture with regard to the freedoms it grants senders, as well as the fixed relationship between mailbox and address, are the critical factors that provide opportunity to those who would abuse it.

Consistent with the Internet's cultural identity of openness and emphasis on personal responsibility, Internet mail grants users the right to send to any addressable mailbox; if an address is known, anyone can address mail to it. Although the act of sending a message is, in essence, an imposition on the recipient, responsibility for the content and quantity of mail sent is the sole dominion of the sender. The benefits of mail service notwithstanding, a mailbox is a privilege not without obligation, as individuals have no say over the receipt of inbound mail. Filtering offers a partial solution, but it cannot compensate for these properties of the underlying architecture.

Moreover, once junk mailers or any unwelcome source gets hold of an address, especially if they distribute it, there is no escape from the inevitable, short of abandoning the address. In general, Internet mail service consists of two distinct entities, the

mailbox and the address. The mailbox is an open container into which delivered mail is placed, while the purpose of an address is to provide its associated mailbox visibility on the network and represent insert privilege to senders. The powerlessness that users experience when faced with junk mail is rooted in the singular, fixed relationship between regular addresses and their corresponding mailboxes, where mailbox and address are often considered indistinct. This suggests the need for a plurality of addresses associated with each mailbox.

Proxy addresses empower individuals against unsolicited mail by emphasizing a many-to-one relationship between address and mailbox, as opposed to the traditional notion of one mailbox, one address. More succinctly, proxy addresses act as disposable front ends, or aliases, for regular, fixed addresses, and are allocated and deallocated directly by user command without administrator involvement. Though address aliasing can be achieved in other ways, such as by forwarding from other addresses (e.g., using the .forward file on Unix systems) or through mail handler rules that map multiple addresses to a single mailbox, they lack the purposefulness and manageability of proxy addresses.

The opportunity to have more than one address on a mailbox allows a user to create numerous addresses, one for each contact or purpose. In any situation in which an address is called for, such as on forms or in environments that have a public distribution, the user should offer a proxy address instead of a regular address. For example, proxy addresses are ideal as return addresses on posts to USENET newsgroups, which are scanned for users' mail addresses en masse by those who distribute unsolicited mailings. Should unwanted mail begin to arrive addressed to a proxy, or when it is no longer needed, the user may delete or suspend the address, terminating the intrusion by removing the aforementioned visibility and insert privilege. The prerogative to create an address expressly for a given contact and remove it in the event of abuse lets recipients selectively apportion access rights to their mailboxes on an individual basis.

Proxy addresses also frustrate those who distribute addresses. The user controlled lifetime of proxy addresses deters their keeping in address databases, as bulk mailers are unlikely to want their mailing lists waterlogged by hordes of potentially useless addresses. Further, mail from an unexpected source clearly indicates that a party to whom a proxy address was intentionally given has shared the address. Proxy addresses yield another benefit in that, unlike regular addresses, which often consist of parts that concede a user's name, their opacity (see Section 2) provides a wall of anonymity that masks such information. In

certain on-line activities, such as chat sessions, uncertainty exists when other participants request a mail address, especially if from a child and the address indicates a name or whereabouts.

It should be noted that the cancellation of a proxy address does not pose the inconvenience that canceling a regular address does, since the optimal use of proxy addresses is to have many in circulation, each held by a particular contact or existing for a specific purpose. A regular address is frequently a user's only address, or one of a few, and issued to contacts far and wide. By contrast, when a proxy address is deleted or suspended, it impacts only a specific distribution, the likely provocation for the action, obviating the need to inform numerous contacts of a change. In addition, it often proves difficult to cancel or change a regular address without disturbing the service account with which it is provided, since mail service is usually an adjunct to other services, such as Internet access provision or on-line portal sites.

2. Anatomy of a Proxy Address

A proxy address is a standard Internet mail address [RFC 822], consisting of a local part, a "@" symbol, and a domain part. The principal of a proxy address is the local part, which is the concatenation of a single ampersand character followed by a string of precisely eight randomly generated alphanumeric characters. The ampersand serves to help implementations discern proxy from non-proxy addresses. The latter, called a proxy identifier, represents a unique value that keys a proxy address within an address space of other proxy addresses managed at an installation. In conjunction with the other elements, the identifier forms a proxy address, as in &38K000PL@myu.edu, &2II9V9JH@research.myu.edu, or &DF989M41@myisp.net, for example. Identifiers are not case sensitive; thus, 0000000A and 0000000a, for example, are identical.

Though a proxy address is uniquely described by its identifier within the address space of a single database, implementations are free to allow for any number of address spaces, hence databases, from which proxy addresses are allocated. This document, however, does not stipulate how multiple address spaces at an installation are differentiated in terms of the one from which a newly created proxy address is allocated. The domain name of a user's regular address, however, is an obvious choice.

Despite giving proxy addresses a cryptic appearance, identifiers are randomly generated, because far fewer address combinations would occur in practice if they were user selectable. Machine chosen identifiers, on the other hand, give each value out of the entire set of 36^8 (i.e., eight alphanumeric characters), or some 2.8 trillion, possibilities equal standing. A large, balanced set of addresses significantly reduces the effectiveness of random or brute force attempts to send unwanted mail into a group of proxy

addresses.

The all-zero identifier is a special case. A proxy address with this identifier (e.g., &00000000@myu.edu) is called a null proxy

and owned by an installation's administrator. The all-zero identifier is never assigned to a proxy address allocated by an ordinary user.

In addition to its identifier, a proxy address carries three other attributes: the name of the user account of its owner, its suspension state, and a remark. The user account name string serves to connect the proxy address to its owner's account, which, in turn, holds the regular address to which the proxy is mapped. The suspension attribute is a Boolean value, represented by either the character "0", for false, or active, or "1", for true, or suspended. A proxy address in the suspended state responds to incoming mail as if it were non-existent. The remark, a string of up to 64 characters, is freely set, used for such things as a short, user defined message about the purpose of the proxy address or arbitrary data assigned by the client implementation to help it in the management of multiple proxies. The user has charge over suspension and the remark; the proxy identifier and user account reference are assigned at creation for the lifetime of the proxy address.

3. User Accounts

Each proxy address is associated with a single user account. Association grants the account holder the right to have mail addressed to a proxy directed to the regular address specified by the account's address attribute, as well as permission to view or change the proxy's attributes. A server implementation maintains a database of user accounts, each of which have, principally, an identifying username, a password, and an address attribute. The username and password are arbitrary strings of visible characters, though implementations may determine case sensitivity or impose string length limits or additional character constraints as they see fit. As stated previously, a proxy address references its associated account by citing this username. The address attribute stores a standard Internet mail address, the regular address coupled with the owner's mailbox. The proxy address is translated into this address by implementations during mail processing.

The password prevents the manipulation of proxy addresses belonging to other users. All interactions between client and server take place in authenticated sessions, in which both username and password have been verified. During authentication, a client may transmit either a clear text password or a digest based on the password protection scheme described in Section 6.2.

A user account has two other attributes: the number of proxy addresses currently owned and the maximum number that may be owned.

An error results if the user attempts to create proxy addresses beyond this limit. It is suggested that a reasonable default maximum for most users be in the range of 10 to 20 proxy addresses. Implementations should allow the administrator to adjust this limit

for each user.

4. Mail Delivery

On its way from source to destination, a message is routed through one or more SMTP [RFC 821] servers. At each stop, a server either advances the message along its source route, if specified, or determines the action to take based on the message's recipient address, either delivering, rejecting, or forwarding the message. In the case of a proxy address, whose role is to front a regular address, which, in turn, corresponds to a mailbox, the server consults its proxy address and user account databases to translate the proxy address into the corresponding regular address, on which the installation's normal mail handling decisions are performed. No part of the message data is modified, although a server may prepend its usual Received header field [RFC 822], the "for" sub-field specifying the translated regular address rather than the proxy address. The regular address becomes the message's operative recipient address for purposes of mail processing at the current and subsequent stops. In some cases, servers may choose an action based upon the proxy address itself, rather than performing the translation, such as to forward all incoming proxy addressed mail to another server.

Administrators and users should be aware that a delivery status notification returned to a sender may include a copy of the transmitted message containing Received header fields that cite the regular address translated from the original proxy address. Given that a proxy address should conceal its corresponding regular address from senders, this presents a security risk for proxy address owners. In particular, forwarding is an issue, because messages acquire such marks that include the translated regular address at each forwarding hop. If an error occurs prior to final delivery, the delivery status notification returned to sender would disclose the address. Success notifications returned by recipients on delivery should also be considered.

On the client side, implementations provide users the means with which to allocate and deallocate proxy addresses, and view and change their attributes. For received proxy addressed mail, implementers should be mindful that, despite the plurality of proxy addresses with respect to a regular address, mail is retrieved from a single mailbox and presented to users in the usual ways. Users, though, may find user interface options that display or sort received mail by recipient address, hence proxy address, useful. For outgoing mail, implementations should offer the user a choice as to which proxy address to use as the source address on new messages. For replies, clients may volunteer the recipient proxy

address of the received message as the reply's source address, provided it is a proxy address that the user owns. Ultimately, in support of the aims discussed in Section 1, the onus is on client applications to promote optimal use of proxy addresses through

effective user interface design.

5. Command-Response Interface

To facilitate the management of proxy addresses, this specification defines a distinct command-response interface for use between connected client and server implementations. As its principal mission is the allocation of mail addresses, support for the service is ideally embedded within SMTP [RFC 821] implementations. PMAP services, therefore, are made available through TCP listening port 25, that assigned SMTP. Clients in this context are typically applications executed on personal workstations that render the protocol's functionality for end users. A server implementation, on the other hand, executes on a machine that has sufficient connectivity and processing resources to provide back end services to multiple users. A PMAP server maintains databases of local user accounts and proxy addresses for mail handlers to check when processing mail and clients to access in the management of proxy addresses.

As stated, PMAP support is best provided closely alongside that for SMTP, its command-response interface available as an alternative to that of SMTP. Once a network connection is established, a client chooses to initiate one of either protocol session, issuing the PMAP command instead of the HELO [RFC 821] (or EHLO [RFC 1869]) command to initiate a PMAP session instead of a SMTP session. As the PMAP command is the only extension to the SMTP command space in this document, it is unrecognized in a PMAP session.

Sessions provide the context for interaction between client and server implementations. The client initiates an interaction by sending a command to the server, which carries out the requested task and returns a response upon completion. All tasks are stateless in that they complete within the scope of a single interaction.

Command and response lines are limited to a maximum of 512 characters, including terminator. All PMAP syntactic elements are encoded as 7-bit, US-ASCII text [US-ASCII]. Implementations should observe case independence with respect to all syntactic elements containing alphabetic characters.

5.1 Commands

A command is transmitted as a line of text, consisting of a command verb, any required arguments, where each is preceded by a space character, and a terminating carriage return/linefeed pair:

```
command-line = verb * ( SP argument ) CRLF
```

PMAP defines eight commands for use in a PMAP session: AUTH, NEW, DEL, SUS, REM, STAT, LIST, and DONE.

The AUTH command is used to declare the username and password of the user account with which to bind the session. Except for the DONE command, which ends a PMAP session, it must be issued in a session before any other command can succeed and is illegal once the user is authenticated.

NEW and DEL create and delete proxy addresses, respectively, while SUS and REM modify their suspension states and remarks. STAT retrieves the attributes of either a proxy address or the session's authenticated user account, depending on the presence of an argument. LIST returns the identifiers of the proxy addresses owned by the authenticated user. Lastly, the DONE command is used to close a PMAP session and revert to a SMTP session. As stated, the PMAP command is valid only in a SMTP session.

5.2 Responses

A PMAP session response, like a command, is a line of text terminated by a carriage return/linefeed pair, whose parameters and optional comment are each preceded by a space character:

```
response-line = status-char ( SP error-keyword /
   * ( SP success-parameter ) ) [ SP comment ] [ CRLF
   multi-line-list-reply ] CRLF
```

The comment strings are defined by implementers, with some suggestions in Section 7.

This response definition is specific only to PMAP session commands. The PMAP command, which is used outside of a PMAP session, is exceptional in that, while its success response conforms to the above, its failure response is a three-digit SMTP result code. The DONE command, also, generates responses in either protocol session, though conversely.

The first element of a PMAP response line is the status character, whose value is either a plus symbol ("+"), for success, or a minus symbol ("-"), for failure. A success response may include parameters determined by the foregoing command:

```
success-response-line = "+" * ( SP success-parameter )
   [ SP comment ] CRLF
```

The LIST command is unique in that it returns a multi-line response.

On failure, a PMAP response line contains one of five error keywords that describes the cause of the failure, following the status character:

```
      failure-response-line = "-" SP ( "SYN" / "GEN" / "ID" / "AUTH" /
```

"MAX") [SP comment] CRLF

The SYN error keyword is returned as a result of any syntax or parsing error, such as an unrecognized command or invalid argument. GEN indicates a general error condition, where the implementation is unable to carry out the requested task; for example, an attempt to create a new proxy cannot succeed because of a resource limitation.

The ID error keyword is returnable only from commands that take a proxy identifier as an argument. It indicates that the session's authenticated user does not own the identified proxy, either because it belongs to another user or does not exist. The AUTH keyword is returned when the username or password arguments to the AUTH command do not correspond to those of any account on the server. It may also result if any command other than AUTH or DONE is issued in a session before a successful authentication or if the AUTH command is issued when a user has already been authenticated.

Unlike the other error keywords, MAX is returned as a potential failure response only by the NEW command. It indicates that an attempt to create a new proxy address exceeds the limit allowed for the user account.

6. Command Reference

The error keywords of potential failure responses are given at the end of each command subsection. Only the failures specific to each command or its arguments are given explicitly.

6.1 PMAP

PMAP is the only command defined herein that exists in the SMTP [RFC 821], as opposed to the PMAP, command space. Consequently, the PMAP command is unrecognized in a PMAP session. Valid at any point in a SMTP session where the QUIT [RFC 821] command may be used, the PMAP command signals the server of the client's desire to transition from a SMTP to a PMAP session. It is equivalent to the QUIT command in that context, except that the network connection is maintained from one protocol session to the next. The PMAP command takes no arguments:

command = "PMAP" CRLF

As with the DONE command, its response is generated in one of either protocol session, depending on success or failure. If the PMAP command fails, SMTP remains the principal, and the server returns a standard SMTP failure response with either the 421, 500, or 502 error code:

```
failure-response = ( "421" / "500" / "502" ) [ SP comment ] CRLF
```

Each indicates that PMAP services are unavailable. Specifically, the 500 error code indicates that the PMAP command is unrecognized. Conversely, the command is recognized in the case of 421 and 502. 421 indicates that service is temporarily unavailable, while 502 means that service is unimplemented.

Should the PMAP command succeed, a PMAP session begins immediately and returns a success response:

```
success-response = "+" SP digest-context [ SP comment ] CRLF
```

Its single parameter, a string of 64 randomly generated visible characters, is the digest context for the session, optionally used in authentication. It is unique each session.

6.2 AUTH

With the exception of DONE, no command may succeed in a PMAP session until a user has been authenticated with AUTH command, which takes as arguments the username and password of the account with which to associate the session:

```
command = "AUTH" SP username SP ( password / digest ) CRLF
```

As described below, a digest of the password may be transmitted instead of the password itself. The digest is a string of 16 hexadecimal characters.

If the username argument matches an existing account name on the server and the password argument matches that held by the identified account, the user is authenticated for the session, and a success response without parameters is returned:

```
success-response = "+" [ SP comment ] CRLF
```

Alternatively, the server returns a failure response with the AUTH error keyword:

```
failure-response = "-" SP "AUTH" [ SP comment ] CRLF
```

This failure response is also generated if the AUTH command is used subsequent to a successful authentication in a session.

In general, a security risk exists whenever sensitive data, such as a password, is transmitted in the clear across a network. For a measure of protection, the AUTH command accepts a digest of the password instead of the actual password. The digest is computed using the MD5 algorithm [RFC 1321] from the concatenation of the digest context string, returned as the success response parameter

to the PMAP command, followed by the account password:

digest := md5(digest-context + password)

If the digest received from the client matches that computed in the same manner by the server, the password is verified and the session is authenticated. Otherwise, the AUTH failure response is returned.

This method of password protection effectively conceals the password, given the difficulty in mapping the digest generated by the MD5 algorithm to the original data containing the password or constructing a second data set that would generate the same digest. Furthermore, the digest passed to the AUTH command is valid only in the session in which it was computed, because the digest context returned in the PMAP command success response used in its computation is generated anew each session. Server implementations may offer administrators the option to disallow clear text passwords.

Potential error keywords: SYN, GEN, AUTH

6.3 NEW

The client issues the NEW command to create a new proxy address:

```
command = "NEW" CRLF
```

On success, the server assigns the name of the authenticated user account to the new proxy's account reference attribute. It also assigns the value false to the suspension attribute, which makes the proxy address active, and a blank string to its remark. The success response includes the identifier of the new proxy address as its single parameter:

```
success-response = "+" SP proxy-id [ SP comment ] CRLF
```

The count of owned proxy addresses belonging to the authenticated user account is incremented on success. User accounts track the number of proxies they own in addition to the maximum number they may own. If an attempt to create a new proxy would exceed this maximum, a failure response with the MAX error keyword is returned:

```
failure-response = "-" SP "MAX" [ SP comment ] CRLF
```

Potential error keywords: SYN, GEN, AUTH, MAX

6.4 DEL

The option to remove an address to cut off unwanted mail is the rationale behind proxy addresses. This is accomplished using the DEL command, which the client issues with a single argument, the identifier of the proxy address to delete:

```
      command = "DEL" SP proxy-id CRLF
```

If the referenced proxy is not owned by the authenticated user, a failure response with the ID error keyword is returned:

```
failure-response = "-" SP "ID" [ SP comment ] CRLF
```

A successful deletion returns a success response without parameters:

```
success-response = "+" [ SP comment ] CRLF
```

The removal of a proxy address decrements the count of owned proxies in the authenticated user account. Any mail sent to the deleted proxy address is rejected by the mail handler with a standard delivery status notification, describing the proxy as an unknown address. (See Section 4.)

Potential error keywords: SYN, GEN, ID, AUTH

6.5 SUS

A suspended proxy address rejects all incoming mail as if the address did not exist, mimicking the affects of deleting a proxy without actually doing so. The client can toggle the suspension state of a proxy address using the SUS command:

```
command = "SUS" SP proxy-id CRLF
```

On success, if the proxy was already suspended, it is made active; if already active, it is suspended. A success response is returned without parameters:

```
success-response = "+" [ SP comment ] CRLF
```

If the proxy address whose identifier was passed as an argument is not owned by the authenticated user account, the suspension state remains unchanged, and a failure response is returned with the ID error keyword:

```
failure-response = "-" SP "ID" [ SP comment ] CRLF
```

A client can determine the current state of the suspension attribute using the STAT command.

Potential error keywords: SYN, GEN, ID, AUTH

6.6 REM

Each proxy address carries a remark attribute, an arbitrary string of characters assigned by the client implementation or user.

Clients, for example, may store custom data to help manage multiple proxy addresses, while users may assign a description of the

purpose of a proxy. While the STAT command is used to retrieve the remark, the REM command assigns its value:

```
command = "REM" SP proxy-id SP remark CRLF
```

The remark string consists of at most 64 non-control characters. If the remark argument contains at least one space, it is enclosed in double quotes. Double quote and backslash characters are, therefore, escaped. To empty the remark attribute, a blank string may be assigned, expressed using two consecutive double quotes.

A successful assignment returns a success response without parameters:

```
success-response = "+" [ SP comment ] CRLF
```

The REM command fails with the ID error keyword if passed an identifier of a proxy address not owned by the authenticated user account:

```
failure-response = "-" SP "ID" [ SP comment ] CRLF
```

As with any other submission by the client, the REM command may fail due to a syntax error, to which it is particularly susceptible given the constraints on the remark argument.

The remark attribute remains unchanged on failure.

Potential error keywords: SYN, GEN, ID, AUTH

6.7 STAT

The STAT command returns the pertinent attributes of either the authenticated user account or a given proxy address. It is issued with or without a proxy identifier as an argument:

```
command = "STAT" [ SP proxy-id ] CRLF
```

Without the argument, a success response is returned with three parameters corresponding to the authenticated user account:

```
account-success-response = "+" SP mailbox SP owned-proxies
   SP max-proxies [ SP comment ] CRLF
```

The parameters contain the traditional mailbox address [RFC 822], to which mail addressed to any proxy address associated with the account is directed, the current number of proxy addresses owned by the authenticated user account, and the maximum number that may be owned under the account.

Alternatively, the STAT command returns two attributes belonging to the proxy address specified by the identifier in the argument:

```
proxy-success-response = "+" SP suspended SP remark
   [ SP comment ] CRLF
```

The suspended and remark parameters give the proxy address's suspension and remark attributes, respectively. If the remark contains at least one space or is empty, it is enclosed in double quotes.

If STAT is issued with the argument, it may fail with the ID error keyword:

```
failure-response = "-" SP "ID" [ SP comment ] CRLF
```

Potential error keywords: SYN, GEN, ID, AUTH

6.8 LIST

A client uses the LIST command to retrieve a listing of identifiers of the proxy addresses associated with the authenticated user account. It is issued without arguments:

```
command = "LIST" CRLF
```

The LIST command, like any other, is susceptible to failure by syntax, general, or authentication error:

```
failure-response = "-" SP ( "SYN" / "GEN" / "AUTH" )
   [ SP comment ] CRLF
```

The success response to the LIST command, however, is unlike that of the other commands. Following the usual success response line, which consists of the plus symbol and optional comment string, a sequence of lines containing one proxy identifier each is returned:

```
success-response = "+" [ SP comment ] * ( CRLF proxy-id ) CRLF
```

The client should query the server using the STAT command to obtain the number of proxy addresses owned by the authenticated user account, hence the number of lines to expect, prior to issuing LIST. The server is not obliged to order the listing.

Potential error keywords: SYN, GEN, AUTH

6.9 DONE

The client closes a PMAP session using the DONE command, which initiates a SMTP session as if a new connection had been established. To close the connection, the client issues the QUIT [RFC 821] command after the transition to a SMTP session. Unlike

the other PMAP session commands, DONE may be used even if the session has not been authenticated with the AUTH command.

```
command = "DONE" CRLF
```

As with the PMAP command, the source of the response depends on success or failure and the switch between protocols. On success, control reverts to SMTP, which returns a standard success response with the 220 greeting code:

```
success-response = "220" [ SP comment ] CRLF
```

Following this command, the client must begin a new PMAP session and re-authenticate in order to use PMAP session commands.

The DONE command, properly phrased, must not fail. However, as with any other submission, the client must cope with the possibility of a syntax error:

```
failure-response = "-" SP "SYN" [ SP comment ] CRLF
```

Potential error keywords: SYN

7. Example Sessions

Failure opening a PMAP session:

```
<connection opened>

S: 220 myu.edu SMTP service ready

C: PMAP

S: 500 Command unrecognized
  -or-
S: 502 Service unimplemented
  -or-
S: 421 Service temporarily unavailable

C: QUIT
S: 221 myu.edu SMTP service closing connection

<connection closed>
```

Success opening a PMAP session, failed authentication:

```
<connection opened>

S: 220 myu.edu SMTP service ready

C: PMAP
S: +
```

```
H/29X)^+CM03/XBNJ%912!\CL66"9MS03);AD872}NS@82L::J97\P50(1J9.W9W
myu.edu PMAP service ready
```

```
      C: AUTH dvader luke
      S: - AUTH Username, password, or digest unrecognized

      C: DONE
      S: 220 myu.edu SMTP service ready

      C: QUIT
      S: 221 myu.edu SMTP service closing connection

      <connection closed>
```

Success opening and authenticating a PMAP session, with representative uses and misuses of commands:

```
      <connection opened>

      S: 220 myu.edu SMTP service ready

      C: PMAP
      S: +
      MV903,A>M677.0&~LF$A0#.39F??=JHG+HL?1K*{NM&!2KE[916!!J1MD0%[88EQ
      myu.edu PMAP service ready

      C: PMAP
      S: - SYN Syntax error

      C: DEL M09Z7812
      S: - AUTH Command used out of context

      C: AUTH hsolo leia
      S: + User authenticated in session

      C: AUTH hsolo leia

      S: - AUTH Command used out of context

      C: NEW
      S: + J779A01P New proxy address created

      C: NEW
      S: - MAX Proxy address not created, ownership limit exceeded

      C: DEL J779A01P
      S: + Proxy address deleted

      C: DEL J779A01P
      S: - ID Proxy address identifier unrecognized

      C: SUS KJD09M09
```

```
S: + Proxy address suspension state toggled
```

```
C: REM KJD09M09 "Imperial newsletter"
S: + Proxy address remark assigned

C: STAT KJD09M09
S: + 1 "Imperial newsletter" Proxy address attributes

C: STAT 2AA09552
S: + 0 "" Proxy address attributes

C: STAT
S: + han@cin.myu.edu 3 4 User account attributes

C: LIST
S: + Owned proxy addresses
S: KJD09M09
S: 2AA09552
S: M09Z7812

C: DONE
S: 220 myu.edu SMTP service ready

C: QUIT
S: 221 myu.edu SMTP service closing connection

<connection closed>
```

8. Syntax Rules

The syntax rules, expressed using Augmented Backus-Naur Form [RFC 2234], describe the elements of the PMAP command-response interface. The elements are encoded as 7-bit, US-ASCII text [US-ASCII]. PMAP specific rules are given in lower case, while those from the set of core rules defined in Section 6.1 of [RFC 2234] are in upper case. The following rules are common to both the command and response sections.

```
alphanum = ALPHA / DIGIT

proxy-id = 8 alphanum

remark = 1*64 VCHAR / DQUOTE *64 vqchar DQUOTE

vqchar = "\" DQUOTE / "\\" / %d32-33 / %d35-91 / %d93-126
```

8.1 Commands

```
command = ( open-session / authenticate / new-proxy /
   delete-proxy / toggle-suspension / set-remark /
   get-account-status / get-proxy-status / list-proxies /
```

```
      close-session ) CRLF

   open-session = "PMAP"  ; valid in SMTP session only
```

```
   authenticate = "AUTH" SP username SP ( password / digest )

   new-proxy = "NEW"

   delete-proxy = "DEL" SP proxy-id

   toggle-suspension = "SUS" SP proxy-id

   set-remark = "REM" SP proxy-id SP remark

   get-account-status = "STAT"

   get-proxy-status = "STAT" SP proxy-id

   list-proxies = "LIST"

   close-session = "DONE"

   username = 1* VCHAR

   password = 1* VCHAR

   digest = 16 HEXDIG
```

8.2 Responses

```
   response = ( success / failure ) CRLF

   success = ( "+" ( open-session-success / authenticate-success /
      new-proxy-success / delete-proxy-success /
      toggle-suspension-success / set-remark-success /
      get-account-status-success / get-proxy-status-success /
      list-proxies-success ) ) / close-session-success

   failure = ( open-session-failure / ( "-"
      SP ( authenticate-failure / new-proxy-failure /
      delete-proxy-failure / toggle-suspension-failure /
      set-remark-failure / get-account-status-failure /
      get-proxy-status-failure / list-proxies-failure /
      close-session-failure ) ) ) [ SP comment ]

   open-session-success = SP digest-context [ SP comment ]

   open-session-failure = "421" / "500" / "502"
      ; generated in SMTP session

   authenticate-success = [ SP comment ]

   authenticate-failure = syntax-error / general-error /
      authentication-error
```

```
   new-proxy-success = SP proxy-id [ SP comment ]

   new-proxy-failure = syntax-error / general-error /
      authentication-error / proxy-limit-error

   delete-proxy-success = [ SP comment ]

   delete-proxy-failure = syntax-error / general-error /
      identifier-error / authentication-error

   toggle-suspension-success = [ SP comment ]

   toggle-suspension-failure = syntax-error / general-error /
      identifier-error / authentication-error

   set-remark-success = [ SP comment ]

   set-remark-failure = syntax-error / general-error /
      identifier-error / authentication-error

   get-account-status-success = SP mailbox SP owned-proxies
      SP max-proxies [ SP comment ]

   get-account-status-failure = syntax-error / general-error /
      authentication-error

   get-proxy-status-success = SP suspended SP remark [ SP comment ]

   get-proxy-status-failure = syntax-error / general-error /
      identifier-error / authentication-error

   list-proxies-success = [ SP comment ] * ( CRLF proxy-id )

   list-proxies-failure = syntax-error / general-error /
      authentication-error

   close-session-success = "220" [ SP comment ]
      ; generated in SMTP session

   close-session-failure = syntax-error

   comment = * ncchar

   digest-context = 64 VCHAR

   proxy-id = 8 alphanum

   mailbox = <standard Internet mail address>

   proxies = number
```

```
   max-proxies = number
```

```
suspended = BIT

number = 1* DIGIT

ncchar = SP / VCHAR

syntax-error = "SYN"

general-error = "GEN"

identifier-error = "ID"

authentication-error = "AUTH"

proxy-limit-error = "MAX"
```

9. Security Considerations

Though the authentication process described in Section 6.2 involves the transfer of an account password each session, it is arguable that PMAP is less susceptible to eavesdropping compared with other protocols, such as the Post Office Protocol [RFC 1939], because PMAP is not as essential a service. Connections to POP servers, for example, are many and frequent, given their role, while PMAP sessions constitute only management chores. In any case, users and administrators must face the risks inherent to transmitting passwords as clear text across a network.

The goal of the password protection scheme defined in Section 6.2 involving the MD5 algorithm [RFC 1321] is to prevent eavesdroppers from capturing the password when the client sends it to the server for authentication. Data is vulnerable in this context specifically when in transit, so it makes sense to transmit it or a representation of it in a scrambled form, hence sending a digest in place of a true password. The procedure requires that both sender and receiver compute the digest in the same way. In addition, the properties of digests generated by the MD5 algorithm practically guarantee that the original data, the password, cannot easily be derived from the digest nor another data set be found that maps to the same digest. For these reasons, this succeeds in protecting the password, since only the two endpoints are expected to know it, thus be able to produce matching digests. Note that digestion is preferable to encryption for these reasons, as well as for its independence from keys and the need to convey them securely.

How proxy addresses might be used in the field poses some security issues, if personal or economic rather than technological. Section 1 outlines the benefits of proxy addresses, notably their plurality, transience, and opacity. As with any tool, its benefits

may be used to both positive and negative ends. In the same way a user may remove a proxy address to stop intrusive mail, a wrongdoer

may hide behind a proxy address in the commission of an offense, with the freedom to delete it at any time to eliminate that recourse. Similarly, as users may feel secure in the anonymity that the code like appearance of proxy addresses provides, offenders may take the same advantage. Fraud, for example, could be committed behind a proxy address.

In all cases, caution should be exercised with the same sensibilities in virtual settings as in the real world. No competent vendor, for example, would accept only a postal box address or a telephone number in lieu of payment for an item, nor would someone make a close acquaintance without substantive information. Likewise, such prudence should be observed by anyone in a tangible personal or economic relationship with someone offering only a proxy address. Implementers are encouraged to provide server based auditing facilities to help track abuse when it occurs.

10. References

[RFC 821] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, August 1982.

[RFC 822] Crocker, D., "Standard for the format of ARPA Internet text messages", STD 11, RFC 822, August 1982.

[RFC 1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC 1869] Klensin, J., et al, "SMTP Service Extensions", STD 10, RFC 1869, November 1995.

[RFC 1939] Myers, J., and Rose, M., "Post Office Protocol, Version 3", STD 53, RFC 1939, May 1996.

[RFC 2234] Crocker, M., "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[US-ASCII] ANSI X3.4:1986, "Coded Character Sets: 7 Bit American National Standard Code for Information Interchange (7-bit ASCII)".

11. Contact Information

Derek Coulter
E-mail: dcoulter@cgo.wave.ca