Internet Draft
draft-ietf-msgtrk-mtqp-08.txt
Valid for six months

T. Hansen
AT&T Laboratories

May 19, 2002

Message Tracking Query Protocol

<draft-ietf-msgtrk-mtqp-08.txt>

Authors' version: 1.18

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This memo and its companions are discussed on the MSGTRK working group mailing list, ietf-msgtrk@imc.org. To subscribe, send a message with the word "subscribe" in the body (on a line by itself) to the address ietf-msgtrk-request@imc.org. An archive of the mailing list may be found at http://www.ietf.org/archive/msgtrk.

Copyright Notice

Abstract

Customers buying enterprise message systems often ask: Can I track the messages? Message tracking is the ability to find out the path that a particular message has taken through a messaging system and the current routing status of that message. This document describes the

Message Tracking Query Protocol that is used in conjunction with extensions to the ESMTP protocol to provide a complete message tracking solution for the Internet.

1. Introduction

The Message Tracking Models and Requirements document [DRAFT-TRACK-MODEL] discusses the models that message tracking solutions could follow, along with requirements for a message tracking solution that can be used with the Internet-wide message infrastructure. This memo and its companions, [DRAFT-TRACK-ESMTP] and [DRAFT-TRACK-TSN], describe a complete message tracking solution that satisfies those requirements. The memo [DRAFT-TRACK-ESMTP] defines an extension to the SMTP service that provides the information necessary to track messages. This memo defines a protocol that can be used to query the status of messages that have been transmitted on the Internet via SMTP. The memo [DRAFT-TRACK-TSN] describes the message/tracking-status [RFC-MIME] media type that is used to report tracking status information. Using the model document's terminology, this solution uses active enabling and active requests with both request and chaining referrals.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC-KEYWORDS].

All syntax descriptions use the ABNF specified by [RFC-ABNF]. Terminal nodes not defined elsewhere in this document are defined in [RFC-ABNF], [RFC-URI], [DRAFT-TRACK-ESMTP] or [RFC-SMTPEXT].

1.2. Changes Made for -07

Added hostname to STARTTLS registration information. Corrected ABNF for STARTTLS.

1.3. Changes Made for -06

Added opt-parameter to STARTTLS and description.

1.4. Changes Made for -05

STARTTLS error response changed from "/unsupported" to "/unavailable".

Fixed some minor nits in the examples and some typos.

1.5. Changes Made for -04

Reworked the SRV lookup description.

Other comments from the list.

Changes to the ABNF.

Changed "must" to "MUST" in section 4.

Changed "may" to "MAY" in section 4.

More examples.

Eliminated the registry of vnd. options.

Eliminated lots of unused references.

1.6. Changes Made for -03

Changed references.

Worked on error codes.

Made examples more real with secrets and hashes.

Fixes to examples.

Added dot-stuffed example.

Additional TLS info.

Better Security Considerations section.

1.7. Changes Made for -02

This section will be removed before publication.

Provided information on lookup for an MTQP server: SRV MTQP, then MX, then A.

Provided a section on firewall considerations

Provided a section on service DNS considerations

At IANA's request, left the port number as XXXX and added more information on the option registry.

Added text on various error conditions and fixed ABNF for error response codes.

Fleshed out the tracking examples.

2. Basic Operation

The Message Tracking Query Protocol (MTQP) is similar to many other line-oriented Internet protocols, such as [POP3] and [NNTP]. Initially, the server host starts the MTQP service by listening on TCP port XXXX (TBD by IANA).

When an MTQP client wishes to make use of the message tracking service, it establishes a TCP connection with the server host, as recorded from the initial message submission or as returned by a previous tracking request. To find the server host, the MTQP client first does an SRV lookup for the server host using DNS SRV records, with a service name of "mtqp" and a protocol name of "tcp", as in _mtqp._tcp.smtp3.example.com. (See the "Usage rules" section in [RFC-SRV] for details.) If the SRV records do not exist, the MTQP client then does an address record lookup for the server host.

When the connection is established, the MTQP server sends a greeting. The MTQP client and MTQP server then exchange commands and responses (respectively) until the connection is closed or aborted.

2.1. Tracking Service DNS Considerations

Because of the ways server host lookups are performed, many different tracking server host configurations are supported.

A mail system that uses a single mail server host and has the MTQP server host on the same server host will most likely have a single MX record pointing at the server host, and if not, will have an address record. Both mail and MTQP clients will access that host directly.

A mail system that uses a single mail server host, but wants tracking queries to be performed on a different machine, MUST have an SRV MTQP record pointing at that different machine.

A mail system that uses multihomed mail servers has two choices for providing tracking services: either all mail servers must be running tracking servers that are able to retrieve information on all messages, or the tracking service must be performed on one (or more) machine(s) that are able to retrieve information on all messages. In the former case, no additional DNS records are needed beyond the MX records already in place for the mail system. In the latter case, SRV MTQP records are needed that point at the machine(s) that are running the tracking service. In both cases, note that the tracking service MUST be able to handle the queries for all messages accepted by that mail system.

2.2. Commands

Commands in MTQP consist of a case-insensitive keyword, possibly followed by one or more parameters. All commands are terminated by a CRLF pair. Keywords and parameters consist of printable ASCII characters. Keywords and parameters are separated by whitespace (one or more space or tab characters). A command line is limited to 998 characters before the CRLF.

2.3. Responses

Responses in MTQP consist of a status indicator that indicates success or failure. Successful commands may also be followed by additional lines of data. All response lines are terminated by a CRLF pair and are limited to 998 characters before the CRLF. There are several status indicators: "+OK" indicates success; "+OK+" indicates a success followed by additional lines of data, a multi-line success response; "-TEMP" indicates a temporary failure; "-ERR" indicates a permanent failure; and "-BAD" indicates a protocol error (such as for unrecognized commands).

A status indicator MAY be followed by a series of machine-parsable, case-insensitive response information giving more data about the errors. These are separated from the status indicator and each other by a single slash character ("/", decimal code 47). Following that, there MAY be white space and a human-readable text message. The human-readable text message is not intended to be presented to the end user, but should be appropriate for putting in a log for use in debugging problems.

In a multi-line success response, each subsequent line is terminated by a CRLF pair and limited to 998 characters before the CRLF. When all lines of the response have been sent, a final line is sent consisting of a single period (".", decimal code 046) and a CRLF pair. If any line of the multi-line response begins with a period, the line is "dot-stuffed" by prepending the period with a second period. When examining a multi-line response, the client checks to see if the line begins with a period. If so, and octets other than CRLF follow, the first octet of the line (the period) is stripped away. If so, and if CRLF immediately follows the period, then the response from the MTQP server is ended and the line containing the ".CRLF" is not considered part of the multi-line response.

An MTQP server MUST respond to an unrecognized, unimplemented, or syntactically invalid command by responding with a negative -BAD status indicator. A server MUST respond to a command issued when the session is in an incorrect state by responding with a negative -ERR status indicator.

2.4. Optional Timers

An MTQP server MAY have an inactivity autologout timer. Such a timer MUST be of at least 10 minutes in duration. The receipt of any command from the client during that interval should suffice to reset the autologout timer. An MTQP server MAY limit the number of commands, unrecognized commands, or total connection time, or MAY use other criteria, to prevent denial of service attacks.

2.5. Firewall Considerations

A firewall mail gateway has two choices when receiving a tracking query for a host within its domain: it may return a response to the query that says the message has been passed on, but no further information is available; or it may perform a chaining operation itself, gathering information on the message from the mail hosts behind the firewall, and returning to the MTQP client the information for each behind-the-firewall hop, or possibly just the final hop information, possibly also disguising the names of any hosts behind the firewall. Which option is picked is an administrative decision and is not further mandated by this document.

3. Initialization and Option Response

Once the TCP connection has been opened by an MTQP client, the MTQP server issues an initial status response that indicates its readiness. If the status response is positive (+OK or +OK+), the client may proceed with other commands.

The initial status response MUST include the response information "/MTQP". Negative responses MUST include a reason code as response information. The following reason codes are defined here; unrecognized reason codes added in the future may be treated as equivalent to "unavailable".

"/" "unavailable"
"/" "admin"

The reason code "/admin" SHOULD be used when the service is unavailable for administrative reasons. The reason code "/unavailable" SHOULD be used when the service is unavailable for other reasons.

If the server has any options enabled, they are listed as the multi-line response of the initial status response, one per line. An option specification consists of an identifier, optionally followed by option-specific parameters. An option specification may be continued onto additional lines by starting the continuation lines with white space. The option identifier is case insensitive. Option identifiers beginning with the characters "vnd." are reserved for vendor use. (See

below.)

One option specification is defined here:

STARTTLS

This capability MUST be listed if the optional STARTTLS command is supported by the MTQP server. It has no parameters.

Example #1 (no options):
S: +OK/MTQP MTQP server ready

Example #2 (service temporarily unavailable):
S: -TEMP/MTQP/admin Service down for admin, call back later

Example #3 (service permanently unavailable):
S: -ERR/MTQP/unavailable Service down

Example #4 (alternative for no options):
S: +OK+/MTQP MTQP server ready
S: .

Example #5 (options available):
S: +OK+/MTQP MTQP server ready
S: starttls
S: vnd.com.example.option2 with parameters private to example.com
S: vnd.com.example.option3 with a very long
S: list of parameters
S: .

4. TRACK Command

Syntax:
"TRACK" 1*WSP envid 1*WSP mtrk-secret CRLF

mtrk-secret = base64

Envid is defined in [DRAFT-TRACK-ESMTP]. Mtrk-secret is the secret A described in [DRAFT-TRACK-ESMTP], encoded using base64.

When the client issues the TRACK command, and the user is validated, the MTQP server retrieves tracking information about an email message. To validate the user, the value of mtrk-secret is hashed using SHA1, as described in [RFC-SHA1]. The hash value is then compared with the value passed with the message when it was originally sent. If the hash values match, the user is validated.

A successful response MUST be multi-line, consisting of a [RFC-

MIME] body part. The MIME body part MUST be of type multipart/related, with subparts of message/tracking-status, as defined in [DRAFT-TRACK-TSN]. The response contains the tracking information about the email message that used the given tracking-id.

In each of the examples below, the envid is "<12345-20010101@example.com>", the secret A is "abcdefgh", and the SHA1 hash B is (in hex) "734ba8b31975d0dbae4d6e249f4e8da270796c94". The message came from example.com and the MTQP server is example2.com.

Example #6 Message Delivered:

```
C: TRACK <12345-20010101@example.com> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S: Content-Type: multipart/related; boundary=%%%%; type=tracking-status
S:
S: --%%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon,  1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user1@example1.com
S: Final-Recipient: rfc822; user1@example1.com
S: Action: delivered
S: Status: 2.5.0
S:
S: --%%%%--
S: .
```

Example #7 Message Transferred:

```
C: TRACK <12345-20010101@example.com> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S: Content-Type: multipart/related; boundary=%%%%; type=tracking-status
S:
S: --%%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon,  1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user1@example1.com
S: Final-Recipient: rfc822; user1@example1.com
S: Action: transferred
S: Remote-MTA: dns; example3.com
S: Last-Attempt-Date: Mon,  1 Jan 2001 19:15:03 -0500
```

```
S: Status: 2.4.0
S:
S: --%%%%--
S: .
```

Example #8 Message Delayed and a Dot-Stuffed Header:

```
C: TRACK <12345-20010101@example.com> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S: Content-Type: multipart/related; boundary=%%%%; type=tracking-status
S: ..Dot-Stuffed-Header: as an example
S:
S: --%%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon,  1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user1@example1.com
S: Final-Recipient: rfc822; user1@example1.com
S: Action: delayed
S: Status: 4.4.1 (No answer from host)
S: Remote-MTA: dns; example3.com
S: Last-Attempt-Date: Mon,  1 Jan 2001 19:15:03 -0500
S: Will-Retry-Until: Thu,  4 Jan 2001 15:15:15 -0500
S:
S: --%%%%--
S: .
```

Example #9 Two Users, One Relayed, One Failed:

```
C: TRACK <12345-20010101@example.com> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S: Content-Type: multipart/related; boundary=%%%%; type=tracking-status
S:
S: --%%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon,  1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user1@example1.com
S: Final-Recipient: rfc822; user1@example1.com
S: Action: relayed
S: Status: 2.1.9
S: Remote-MTA: dns; example3.com
S: Last-Attempt-Date: Mon,  1 Jan 2001 19:15:03 -0500
S:
```

```
S: Original-Recipient: rfc822; user2@example1.com
S: Final-Recipient: rfc822; user2@example1.com
S: Action: failed
S: Status 5.2.2 (Mailbox full)
S: Remote-MTA: dns; example3.com
S: Last-Attempt-Date: Mon,  1 Jan 2001 19:15:03 -0500
S:
S: --%%%%--
S: .
```

Example #10 Firewall:

```
C: TRACK <12345-20010101@example.com> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S: Content-Type: multipart/related; boundary=%%%%; type=tracking-status
S:
S: --%%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon,  1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user1@example1.com
S: Final-Recipient: rfc822; user1@example1.com
S: Action: relayed
S: Status: 2.1.9
S: Remote-MTA: dns; smtp.example3.com
S: Last-Attempt-Date: Mon,  1 Jan 2001 19:15:03 -0500
S:
S: --%%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; smtp.example3.com
S: Arrival-Date: Mon,  1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user2@example1.com
S: Final-Recipient: rfc822; user4@example3.com
S: Action: delivered
S: Status: 2.5.0
S:
S: --%%%%--
S: .
```

Example #11 Firewall, Combining Per-Recipient Blocks:

```
C: TRACK <12345-20010101@example.com> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S: Content-Type: multipart/related; boundary=%%%%; type=tracking-status
```

```
S:
S: --%%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon,  1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user1@example1.com
S: Final-Recipient: rfc822; user1@example1.com
S: Action: relayed
S: Status: 2.1.9
S: Remote-MTA: dns; smtp.example3.com
S: Last-Attempt-Date: Mon,  1 Jan 2001 19:15:03 -0500
S:
S: Original-Recipient: rfc822; user2@example1.com
S: Final-Recipient: rfc822; user4@example3.com
S: Action: delivered
S: Status: 2.5.0
S:
S: --%%%%--
S: .
```

Example #12 Firewall, Hiding System Names Behind the Firewall:

```
C: TRACK <12345-20010101@example.com> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S: Content-Type: multipart/related; boundary=%%%%; type=tracking-status
S:
S: --%%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon,  1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user1@example1.com
S: Final-Recipient: rfc822; user1@example1.com
S: Action: relayed
S: Status: 2.1.9
S: Remote-MTA: dns; example2.com
S: Last-Attempt-Date: Mon,  1 Jan 2001 19:15:03 -0500
S:
S: --%%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon,  1 Jan 2001 15:15:15 -0500
```

```
S:
S: Original-Recipient: rfc822; user2@example1.com
S: Final-Recipient: rfc822; user4@example1.com
S: Action: delivered
S: Status: 2.5.0
S:
S: --%%%%--
S: .
```

5. COMMENT Command

Syntax:

```
"COMMENT" opt-text CRLF

opt-text = [WSP *(VCHAR / WSP)]
```

When the client issues the COMMENT command, the MTQP server MUST respond with a successful response (+OK or +OK+). All optional text provided with the COMMENT command are ignored.

6. STARTTLS Command

Syntax:

```
"STARTTLS" [WSP hostname] CRLF
```

TLS [TLS], more commonly known as SSL, is a popular mechanism for enhancing TCP communications with privacy and authentication. An MTQP server MAY support TLS. If an MTQP server supports TLS, it MUST include "STARTTLS" in the option specifications list on protocol startup.

The optional parameter, if specified, MUST be a fully qualified domain name. A client MAY specify the hostname it believes it is speaking with so that the server may respond with the proper TLS certificate. This is useful for virtual servers that provide message tracking for multiple domains (i.e., virtual hosting).

If the server returns a negative response, it MAY use one of the following response codes:

"/" "unsupported"
"/" "unavailable"
"/" "tlsinprogress"

If TLS is not supported, then a response code of "/unsupported" SHOULD be used. If TLS is not available for some other reason, then a response code of "/unavailable" SHOULD be used. If a TLS session is already in progress, then it is a protocol error and "-BAD" MUST be returned with a response code of "/tlsinprogress".

After receiving a positive response to a STARTTLS command, the client MUST start the TLS negotiation before giving any other MTQP commands.

If the MTQP client is using pipelining (see below), the STARTTLS command must be the last command in a group.

6.1. Processing After the STARTTLS Command

If the TLS handshake fails, the server SHOULD abort the connection.

After the TLS handshake has been completed, both parties MUST immediately decide whether or not to continue based on the authentication and privacy achieved. The MTQP client and server may decide to move ahead even if the TLS negotiation ended with no authentication and/or no privacy because most MTQP services are performed with no authentication and no privacy, but some MTQP clients or servers may want to continue only if a particular level of authentication and/or privacy was achieved.

If the MTQP client decides that the level of authentication or privacy is not high enough for it to continue, it SHOULD issue an MTQP QUIT command immediately after the TLS negotiation is complete. If the MTQP server decides that the level of authentication or privacy is not high enough for it to continue, it SHOULD reply to every MTQP command from the client (other than a QUIT command) with a negative "-ERR" response and a response code of "/insecure".

6.2. Result of the STARTTLS Command

Upon completion of the TLS handshake, the MTQP protocol is reset to the initial state (the state in MTQP after a server starts up). The server MUST discard any knowledge obtained from the client prior to the TLS negotiation itself. The client MUST discard any knowledge obtained from the server, such as the list of MTQP options, which was not obtained from the TLS negotiation itself.

At the end of the TLS handshake, the server acts as if the connection had been initiated and responds with an initial status response and, optionally, a list of server options. The list of MTQP server options received after the TLS handshake MUST be different than the list returned before the TLS handshake. In particular, a server MUST NOT return the STARTTLS option in the list of server options after a TLS handshake has completed.

Both the client and the server MUST know if there is a TLS session active. A client MUST NOT attempt to start a TLS session if a TLS session is already active.

7. QUIT Command

Syntax:
"QUIT" CRLF

When the client issues the QUIT command, the MTQP session terminates. The QUIT command has no parameters. The server MUST respond with a successful response. The client MAY close the session from its end immediately after issuing this command (if the client is on an operating system where this does not cause problems).

8. Pipelining

The MTQP client may elect to transmit groups of MTQP commands in batches without waiting for a response to each individual command. The MTQP server MUST process the commands in the order received.

Specific commands may place further constraints on pipelining. For example, STARTTLS must be the last command in a batch of MTQP commands.

The following two examples are identical:

Example #13 :
C: TRACK <tracking-id> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S:
S: ... tracking details #1 go here ...
S: .
C: TRACK <tracking-id-2> QUJDREVGR0gK
S: +OK+ Tracking information follows
S:
S: ... tracking details #2 go here ...
S: .

Example #14 :
C: TRACK <tracking-id> YWJjZGVmZ2gK
C: TRACK <tracking-id-2> QUJDREVGR0gK
S: +OK+ Tracking information follows
S:
S: ... tracking details #1 go here ...
S: .
S: +OK+ Tracking information follows
S:
S: ... tracking details #2 go here ...
S: .

9. URL Format

The MTQP URL scheme is used to designate MTQP servers on Internet hosts accessible using the MTQP protocol. An MTQP URL takes one of the following forms:

```
mtqp://<mserver>/track/<envid>/<mtrk-secret>
mtqp://<mserver>:<port>/track/<envid>/<mtrk-secret>
```

The first form is used to refer to an MTQP server on the standard port, while the second form specifies a non-standard port. Both of these forms specify that the TRACK command is to be issued using the given tracking id (envid) and authorization secret (mtrk-secret). The path element "/track/" is case insensitive, but the envid and mtrk-secret may not be.

9.1. MTQP URL Syntax

This is an ABNF description of the MTQP URL.

```
mtqp-url = "mtqp://" net_loc "/track/" envid "/" mtrk-secret
```

10. IANA Considerations

System port number XXXX - TBD by IANA

The service name to be registered with the Internet Assigned Number Authority (IANA) is "MTQP".

This document requests that IANA maintain one new registry: MTQP options. The registry's purpose is to register options to this protocol. Options whose names do not begin with "vnd." MUST be defined in a standards track or IESG approved experimental RFC. New MTQP options MUST include the following information as part of their definition:

```
option identifier
option parameters
added commands
standard commands affected
specification reference
discussion
```

One MTQP option is defined in this document, with the following registration definition:

```
option identifier: STARTTLS
option parameters: none
added commands: STARTTLS
```

standard commands affected: none
specification reference: RFC TBD
discussion: see RFC TBD

Additional vendor-specific options for this protocol have names that begin with "vnd.". After the "vnd." would appear the reversed domain name of the vendor, another dot ".", and a name for the option itself. For example, "vnd.com.example.extinfo" might represent a vendor-specific extension providing extended information by the owner of the "example.com" domain. These names MAY be registered with IANA.

11. Security Considerations

If the originator of a message were to delegate his or her tracking request to a third party, this would be vulnerable to snooping over unencrypted sessions. The user can decide on a message-by-message basis if this risk is acceptable.

The security of tracking information is dependent on the randomness of the secret chosen for each message and the level of exposure of that secret. If different secrets are used for each message, then the maximum exposure from tracking any message will be that single message for the time that the tracking information is kept on any MTQP server. If this level of exposure is too much, TLS may be used to reduce the exposure further.

It should be noted that message tracking is not an end-to-end mechanism. Thus, if an MTQP client/server pair decide to use TLS privacy, they are not securing tracking queries with any prior or successive MTQP servers.

Both the MTQP client and server must check the result of the TLS negotiation to see whether acceptable authentication or privacy was achieved. Ignoring this step completely invalidates using TLS for security. The decision about whether acceptable authentication or privacy was achieved is made locally, is implementation-dependent, and is beyond the scope of this document.

The MTQP client and server should note carefully the result of the TLS negotiation. If the negotiation results in no privacy, or if it results in privacy using algorithms or key lengths that are deemed not strong enough, or if the authentication is not good enough for either party, the client may choose to end the MTQP session with an immediate QUIT command, or the server may choose to not accept any more MTQP commands.

A man-in-the-middle attack can be launched by deleting the "STARTTLS" option response from the server. This would cause the client

not to try to start a TLS session. An MTQP client can protect against this attack by recording the fact that a particular MTQP server offers TLS during one session and generating an alarm if it does not appear in an option response for a later session.

If TLS is not used, a tracking request is vulnerable to replay attacks, such that a snoop can later replay the same handshake again to potentially gain more information about a message's status.

Before the TLS handshake has begun, any protocol interactions are performed in the clear and may be modified by an active attacker. For this reason, clients and servers MUST discard any knowledge obtained prior to the start of the TLS handshake upon completion of the TLS handshake.

If a client/server pair successfully performs a TLS handshake and the server does chaining referrals, then the server SHOULD attempt to negotiate TLS at the same security level at the next hop. In a hop-by-hop scenario, STARTTLS is a request for "best effort" security and should be treated as such.

SASL is not used because authentication is per message rather than per user.

12. Protocol Syntax

This is a collected ABNF description of the MTQP protocol.

```
conversation = command-response *( client-command command-response )

# client side
client-command = track-command / starttls-command / quit-command /
comment-command

track-command = "TRACK" 1*WS envid 1*WS mtrk-secret CRLF

mtrk-secret = base64

starttls-command = "STARTTLS" [WSP hostname] CRLF

quit-command = "QUIT" CRLF

comment-command = "COMMENT" opt-text CRLF

# server side
command-response = success-response / temp-response / error-response /
bad-response

temp-response = "-TEMP" response-info opt-text CRLF
```

```
     opt-text = [WSP *(VCHAR / WSP)]
```

```
error-response = "-ERR" response-info opt-text CRLF

bad-response = "-BAD" response-info opt-text CRLF

success-response = single-line-success / multi-line-success

single-line-success = "+OK" response-info opt-text CRLF

multi-line-success = "+OK+" response-info opt-text CRLF *dataline dotcrlf

dataline = *998OCTET CRLF

dotcrlf = "." CRLF

option-list = *option-line

option-line = identifier opt-text *(CRLF WSP opt-text) CRLF

NAMECHAR = ALPHA / DIGIT / "-" / "_"

identifier = (ALPHA / "_") *NAMECHAR)

response-info = *( "/" ( "admin" / "unavailable" / "unsupported" /
        "tlsinprogress" / "insecure" / 1*NAMECHAR ) )
```

13. Acknowledgements

The description of STARTTLS is based on [RFC-SMTP-TLS].

14. References

[RFC-SHA1] RFC TBD, D. Eastlake & P. Jones, "US Secure Hash Standard 1 (SHA1)", TBD 2001.

[RFC-MIME] RFC 2045, N. Freed & N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", Innosoft, First Virtual, November 1996.

[RFC-ABNF] RFC 2234, D. Crocker, Editor, and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", Internet Mail Consortium, Demon Internet Ltd., November 1997.

[RFC-KEYWORDS] RFC 2119, S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", Harvard University, March 1997.

[RFC-SMTPEXT] RFC 2554, J. Myers, "SMTP Service Extension for Authentication", Netscape Communications, March 1999.

[RFC-SMTP-TLS] RFC2487, P. Hoffman, "SMTP Service Extension for Secure SMTP over TLS", Internet Mail Consortium, January 1999.

[RFC-SRV] RFC 2782, A. Gulbrandsen, P. Vixie, L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)" Troll Technologies, Internet Software Consortium, Microsoft Corp., February 2000

[DRAFT-TRACK-ESMTP] draft-ietf-msgtrk-smtpext-*.txt, E. Allman, T. Hansen, "SMTP Service Extension for Message Tracking", Sendmail, Inc., AT&T Laboratories, TBD 2001.

[DRAFT-TRACK-MODEL] draft-ietf-msgtrk-model-*.txt, T. Hansen, "Message Tracking Models and Requirements", AT&T Laboratories, TBD 2001.

[DRAFT-TRACK-TSN] draft-ietf-msgtrk-trkstat-*.txt, E. Allman, "The Message/Tracking-Status MIME Extension", Sendmail, Inc., TBD 2001.

[RFC-URI] RFC 2396, T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", MIT/LCS, U. C. Irvine, Xerox Corporation, August 1998.

15. Author's Address

Tony Hansen
AT&T Laboratories
Middletown, NJ 07748
USA

Phone: +1.732.420.8934
E-Mail: tony@att.com

16. Full Copyright Statement

This document expires November 19, 2002.