

White Paper

The following is a Technical White Paper on Total Mail Defense (hereinafter "TMD") assembled by Ron Edison, Founder and CTO of IDT, and his development team.

Blocking Spam & Mal-ware at the Mail Gateway

ABSTRACT

Today's Internet is not the friendly place it once was.

In bygone days, desktop antivirus and a judicious selection of applications was all most users needed to concern themselves with to achieve safe computing.

Today, however, the avalanche of spam, malware, viruses, trojans, worms, phishing attacks and other security threats have vastly changed this landscape.

So much so, that some users and organizations have resorted to "blacklisted until proven legitimate" messaging strategy. This is unfortunate as it largely defeats the entire purpose of email: fast, easy and accessible communications for users regardless of intervening distance. This approach, while possibly useful for casual email users, is utterly disruptive of an email driven businesses, particularly those that rely on rapid receipt of email from non-predictable email addresses.

Other users resort to limited (and vendor-specific) desktop application software, blocking all mail that matches a limited set of criteria sure to generate false positives or other inflexible or incomplete solutions.

Blocking spam & malware at the mail gateway has emerged as the most effective strategy of securing mail and keeping user inboxes and workstations safe from email-borne threats extant today.

STRATEGY

The trend of email-borne threats continues to rise. With the now 85%+ of all email traffic being spam or worse, corporate mail servers and user workstations more and more feel the brunt of these attacks.

Email content protection on the mail gateway provides protection other solutions cannot.

TMD presents functionality that combines rejection of immediately identifiable spam and other malware before the mail transmission channel is opened (percentages of rejection of 90%+ have been recorded of such mail at this step) with virtually zero false positives (the percentage is far less than 1%) followed by intelligent mail analysis, maximal delivery of legitimate mail to users and minimal administrative overhead for users and administrators.

The fact of this functionality residing on the mail gateway is critical:

- A large percentage of mail can be immediately rejected based on dynamically learned characteristics, saving processing time and bandwidth
- Mail for non-existent users to which huge volumes of spam are often sent are also rejected immediately using a dynamically generated list of valid users from the company mail server

- Gateway-rejected mail requires minuscule computing resources and bandwidth, a tiny fraction of the resources required to accept the message, analyzing it, bounce it, quarantine it etc.
- Email-borne malware is blocked at the gateway by a thorough combination of virus scanning and executable attachment stripping, thus protecting corporate mail servers and workstations, and vastly reducing overhead, management and disinfection time and costs.

Technical Specs of pre-transmission channel establishment email rejection

The first element here is that TMD *rejects* messages **before** the mail transmission channel is opened and **before** the system has expended resources to process, analyze, virus check, attachment check and continue with many other resource intensive operations. This first key step vastly reduces mail volume, bandwidth utilization and generally allows mail systems to do what they were designed to do in the first place -- deliver email. Many other commercial products, appliances, hosted anti-spam services, et al, miss this critical step, so fruitful in rebuffing spam en masse.

As a backdrop to this, spammers have been exploiting these characteristics of the SMTP protocol for years, faking the return path and delivering messages to users that are ultimately nearly impossible to trace. Thus, TMD turns this against them, rejecting huge quantities of mail before the mail gateway expends any resources examining message content.

This rejection occurs in the pre-transmission channel as described in RFC 821/2821.

Quotation of RFC 2821: (<http://www.faqs.org/rfcs/rfc2821.html>)

2.4 General Syntax Principles and Transaction Model SMTP commands and replies have a rigid syntax. All commands begin with a command verb. All Replies begin with a three digit numeric code. In some commands and replies, arguments MUST follow the verb or reply code. Some commands do not accept arguments (after the verb), and some reply codes are followed, sometimes optionally, by free form text. In both cases, where text appears, it is separated from the verb or reply code by a space character. Complete definitions of commands and replies appear in section 4.

Verbs and argument values (e.g., "TO:" or "to:" in the RCPT command and extension name keywords) are not case sensitive, with the sole exception in this specification of a mailbox local-part (SMTP Extensions may explicitly specify case-sensitive elements). That is, a command verb, an argument value other than a mailbox local-part, and free form text MAY be encoded in upper case, lower case, or any mixture of upper and lower case with no impact on its meaning. This is NOT true of a mailbox local-part. The local-part of a mailbox MUST BE treated as case sensitive. Therefore, SMTP implementations MUST take care to preserve the case of mailbox local-parts. Mailbox domains are not case sensitive. In particular, for some hosts the user "smith" is different from the user "Smith". However, exploiting the case sensitivity of mailbox local-parts impedes interoperability and is discouraged.

A few SMTP servers, in violation of this specification (and [RFC 821](#)) require that command verbs be encoded by clients in upper case. Implementations MAY wish to employ this encoding to accommodate those servers.

The argument field consists of a variable length character string ending with the end of the line, i.e., with the character sequence <CRLF>. The receiver will take no action until this sequence is received.

The syntax for each command is shown with the discussion of that command. Common elements and parameters are shown in section 4.1.2.

Commands and replies are composed of characters from the ASCII character set [1]. When the transport service provides an 8-bit byte (octet) transmission channel, each 7-bit character is transmitted right justified in an octet with the high order bit cleared to zero. More specifically, the unextended SMTP service provides seven bit transport only. An originating SMTP client which has not successfully negotiated an appropriate extension with a particular server MUST NOT transmit

messages with information in the high-order bit of octets. If such messages are transmitted in violation of this rule, receiving SMTP servers MAY clear the high-order bit or reject the message as invalid. In general, a relay SMTP SHOULD assume that the message content it has received is valid and, assuming that the envelope permits doing so, relay it without inspecting that content. Of course, if the content is mislabeled and the data path cannot accept the actual content, this may result in ultimate delivery of a severely garbled message to the recipient. Delivery SMTP systems MAY reject ("bounce") such messages rather than deliver them. No sending SMTP system is permitted to send envelope commands in any character set other than US-ASCII; receiving systems SHOULD reject such commands, normally using "500 syntax error - invalid character" replies.

Eight-bit message content transmission MAY be requested of the server by a client using extended SMTP facilities, notably the "8BITMIME" extension [20]. 8BITMIME SHOULD be supported by SMTP servers.

However, it MUST not be construed as authorization to transmit unrestricted eight bit material. 8BITMIME MUST NOT be requested by senders for material with the high bit on that is not in MIME format with an appropriate content-transfer encoding; servers MAY reject such messages.

The metalinguistic notation used in this document corresponds to the "Augmented BNF" used in other Internet mail system documents. The reader who is not familiar with that syntax should consult the ABNF specification [8]. Metalanguage terms used in running text are surrounded by pointed brackets (e.g., <CRLF>) for clarity.

3. The SMTP Procedures: An Overview

This section contains descriptions of the procedures used in SMTP: session initiation, the mail transaction, forwarding mail, verifying mailbox names and expanding mailing lists, and the opening and closing exchanges. Comments on relaying, a note on mail domains, and a discussion of changing roles are included at the end of this section. Some complete scenarios are presented in appendix D.

3.1 Session Initiation

An SMTP session is initiated when a client opens a connection to a server and the server responds with an opening message.

SMTP server implementations MAY include identification of their software and version information in the connection greeting reply after the 220 code, a practice that permits more efficient isolation and repair of any problems. Implementations MAY make provision for SMTP servers to disable the software and version announcement where it causes security concerns. While some systems also identify their contact point for mail problems, this is not a substitute for maintaining the required "postmaster" address (see section 4.5.1).

The SMTP protocol allows a server to formally reject a transaction while still allowing the initial connection as follows: a 554 response MAY be given in the initial connection opening message instead of the 220. A server taking this approach MUST still wait for the client to send a QUIT (see section the connection and SHOULD respond to any intervening commands with "503 bad sequence of commands". Since an attempt to make an SMTP connection to such a system is probably in error, a server returning a 554 response on connection opening SHOULD provide enough information in the reply text to facilitate debugging of the sending system.

3.2 Client Initiation

Once the server has sent the welcoming message and the client has received it, the client normally sends the EHLO command to the server, indicating the client's identity. In addition to opening the session, use of EHLO indicates that the client is able to process service extensions and requests that the server provide a list of the extensions it supports. Older SMTP systems which are unable to support service extensions and contemporary clients which do not require service extensions in the mail session being initiated, MAY use HELO instead of EHLO. Servers MUST NOT return the extended EHLO-style response to a HELO command. For a particular connection attempt, if the server returns a "command not recognized" response to EHLO, the client SHOULD be able to fall back and send HELO. In the EHLO command the host sending the command identifies itself; the command may be interpreted as saying "Hello, I am <domain>" (and, in the case of EHLO, "and I support service extension requests").

3.3 Mail Transactions

There are three steps to SMTP mail transactions. The transaction starts with a MAIL command which gives the sender identification. (In general, the MAIL command may be sent only when no mail transaction is in progress; see section 4.1.4.) A series of one or more RCPT commands follows giving the receiver information. Then a DATA command initiates transfer of the mail data and is terminated by the "end of mail" data indicator, which also confirms the transaction.

The first step in the procedure is the MAIL command.

MAIL FROM:<reverse-path> [SP <mail-parameters>] <CRLF>

This command tells the SMTP-receiver that a new mail transaction is starting and to reset all its state tables and buffers, including any recipients or mail data. The <reverse-path> portion of the first or only argument contains the source mailbox (between "<" and ">" brackets), which can be used to report errors (see section 4.2 for a discussion of error reporting). If accepted, the SMTP server returns a 250 OK reply. If the mailbox specification is not acceptable for some reason, the server MUST return a reply indicating whether the failure is permanent (i.e., will occur again if the client tries to send the same address again) or temporary (i.e., the address might be accepted if the client tries again later). Despite the apparent scope of this requirement, there are circumstances in which the acceptability of the reverse-path may not be determined until one or more forward-paths (in RCPT commands) can be examined. In those cases, the server MAY reasonably accept the reverse-path (with a 250 reply) and then report problems after the forward-paths are received and examined. Normally, failures produce 550 or 553 replies.

Historically, the <reverse-path> can contain more than just a mailbox, however, contemporary systems SHOULD NOT use source routing (see appendix C).

The optional <mail-parameters> are associated with negotiated SMTP service extensions (see section 2.2).

The second step in the procedure is the RCPT command.

RCPT TO:<forward-path> [SP <rcpt-parameters>] <CRLF>

The first or only argument to this command includes a forward-path (normally a mailbox and domain, always surrounded by "<" and ">" brackets) identifying one recipient. If accepted, the SMTP server returns a 250 OK reply and stores the forward-path. If the recipient is known not to be a deliverable address, the SMTP server returns a 550 reply, typically with a string such as "no such user - " and the mailbox name (other circumstances and reply codes are possible). This step of the procedure can be repeated any number of times.

The <forward-path> can contain more than just a mailbox. Historically, the <forward-path> can be a source routing list of hosts and the destination mailbox, however, contemporary SMTP clients SHOULD NOT utilize source routes (see appendix C). Servers MUST be prepared to encounter a list of source routes in the forward path, but SHOULD ignore the routes or MAY decline to support the relaying they imply. Similarly, servers MAY decline to accept mail that is destined for other hosts or

systems. These restrictions make a server useless as a relay for clients that do not support full SMTP functionality. Consequently, restricted-capability clients **MUST NOT** assume that any SMTP server on the Internet can be used as their mail processing (relaying) site. If a RCPT command appears without a previous MAIL command, the server **MUST** return a 503 "Bad sequence of commands" response. The optional <rcpt-parameters> are associated with negotiated SMTP service extensions (see section 2.2).

The third step in the procedure is the DATA command (or some alternative specified in a service extension).

DATA <CRLF>

If accepted, the SMTP server returns a 354 Intermediate reply and considers all succeeding lines up to but not including the end of mail data indicator to be the message text. When the end of text is successfully received and stored the SMTP-receiver sends a 250 OK reply.

Since the mail data is sent on the transmission channel, the end of mail data must be indicated so that the command and reply dialog can be resumed. SMTP indicates the end of the mail data by sending a line containing only a "." (period or full stop). A transparency procedure is used to prevent this from interfering with the user's text (see section 4.5.2).

The end of mail data indicator also confirms the mail transaction and tells the SMTP server to now process the stored recipients and mail data. If accepted, the SMTP server returns a 250 OK reply. The DATA command can fail at only two points in the protocol exchange:

- If there was no MAIL, or no RCPT, command, or all such commands were rejected, the server **MAY** return a "command out of sequence" (503) or "no valid recipients" (554) reply in response to the DATA command. If one of those replies (or any other 5yz reply) is received, the client **MUST NOT** send the message data; more generally, message data **MUST NOT** be sent unless a 354 reply is received.
- If the verb is initially accepted and the 354 reply issued, the DATA command should fail only if the mail transaction was incomplete (for example, no recipients), or if resources were unavailable (including, of course, the server unexpectedly becoming unavailable), or if the server determines that the message should be rejected for policy or other reasons.
- However, in practice, some servers do not perform recipient verification until after the message text is received. These servers **SHOULD** treat a failure for one or more recipients as a "subsequent failure" and return a mail message as discussed in section 6. Using a "550 mailbox not found" (or equivalent) reply code after the data are accepted makes it difficult or impossible for the client to determine which recipients failed.
- When [RFC 822](#) format [7, 32] is being used, the mail data include the memo header items such as Date, Subject, To, Cc, From. Server SMTP systems **SHOULD NOT** reject messages based on perceived defects in the [RFC 822](#) or MIME [12] message header or message body. In particular, they **MUST NOT** reject messages in which the numbers of Resent-fields do not match or Resent-to appears without Resent-from and/or Resent-date.

Mail transaction commands MUST be used in the order discussed above.

After the initial rejection of immediately identifiable illegitimate email, TMD enters it's analysis phase including:

- Multiple RBL Tests

Using high-capacity caching DNS servers, TMD maintains high message delivery performance while utilizing multiple RBL databases

- Mail content and characteristic checks, header, subject and content checking.

Common spammer tactics have been carefully analyzed and messages are run through hundreds of extensive rule-based tests. Most spammers do not write their own code, they use existing header-hiding spamware

There is a back history to this:

- Spamware years ago used identifiable X-Mailer headers and often easily/thusly blocked
- Spammers then wised up and began forging legitimate email client applications
- Careful analysis and comparison of hex encoded timestamps often revealed forgeries
- MIME Structure analysis including MIME boundary patterns
- Detection of use of RDNS in SMTP protocol transactions: HELO/EHLO
- Another spammer technique: fake received headers to avoid detection/tracing in violation of RFC2821 excerpted here:

4.4 Trace Information

When an SMTP server receives a message for delivery or further processing, it MUST insert trace ("time stamp" or "Received") information at the beginning of the message content.

This line MUST be structured as follows:

- ***The FROM field, which MUST be supplied in an SMTP environment, SHOULD contain both (1) the name of the source host as presented in the EHLO command and (2) an address literal containing the IP address of the source, determined from the TCP connection.***
- ***The ID field MAY contain an "@" as suggested in [RFC 822](#), but this is not required.***
- ***The FOR field MAY contain a list of <path> entries when multiple RCPT commands have been given. This may raise some security issues and is usually not desirable; see section 7.2.***

An Internet mail program MUST NOT change a Received: line that was previously added to the message header. SMTP servers MUST prepend Received lines to messages; they MUST NOT change the order of existing lines or insert Received lines in any other location.

As the Internet grows, comparability of Received fields is important for detecting problems, especially slow relays. SMTP servers that create Received fields SHOULD use explicit offsets in the dates (e.g., -0800), rather than time zone names of any type. Local time (with an offset) is referred to UT when feasible. This formulation allows slightly more information about local circumstances to be specified. If UT is needed, the receiver need merely do some simple arithmetic to convert the values. Use of UT loses information about the time zone-location of the server. If it is desired to supply a time zone name, it SHOULD be included in a comment.

When the delivery SMTP server makes the "final delivery" of a message, it inserts a return-path line at the beginning of the mail data. This use of return-path is required; mail systems MUST support it. The return-path line preserves the information in the <reverse-path> from the MAIL command. Here, final delivery means the message has left the SMTP environment. Normally, this would mean it had been delivered to the destination user or an associated mail drop, but in some cases it may be further processed and transmitted by another mail system.

It is possible for the mailbox in the return path to be different from the actual sender's mailbox, for example, if error responses are to be delivered to a special error handling mailbox rather than to the message sender. When mailing lists are involved, this arrangement is common and useful as a means of directing errors to the list maintainer rather than the message originator.

The text above implies that the final mail data will begin with a return path line, followed by one or more time stamp lines. These lines will be followed by the mail data headers and body [32].

It is sometimes difficult for an SMTP server to determine whether or not it is making final delivery since forwarding or other operations may occur after the message is accepted for delivery. Consequently, any further (forwarding, gateway, or relay) systems MAY remove the return path and

rebuild the MAIL command as needed to ensure that exactly one such line appears in a delivered message.

A message-originating SMTP system SHOULD NOT send a message that already contains a Return-path header. SMTP servers performing a relay function MUST NOT inspect the message data, and especially not to the extent needed to determine if Return-path headers are present.

SMTP servers making final delivery MAY remove Return-path headers before adding their own.

The primary purpose of the Return-path is to designate the address to which messages indicating non-delivery or other mail system failures are to be sent. For this to be unambiguous, exactly one return path SHOULD be present when the message is delivered. Systems using RFC 822 syntax with non-SMTP transports SHOULD designate an unambiguous address, associated with the transport envelope, to which error reports (e.g., non-delivery messages) should be sent.

Historical note: Text in [RFC 822](#) that appears to contradict the use of the Return-path header (or the envelope reverse path address from the MAIL command) as the destination for error messages is not applicable on the Internet. The reverse path address (as copied into the Return-path) MUST be used as the target of any mail containing delivery error messages.

In particular:

- a gateway from SMTP->elsewhere SHOULD insert a return-path header, unless it is known that the "elsewhere" transport also uses Internet domain addresses and maintains the envelope sender address separately.

- a gateway from elsewhere->SMTP SHOULD delete any return-path header present in the message, and either copy that information to the SMTP envelope or combine it with information present in the envelope of the other transport system to construct the reverse path argument to the MAIL command in the SMTP envelope.

The server must give special treatment to cases in which the processing following the end of mail data indication is only partially successful. This could happen if, after accepting several recipients and the mail data, the SMTP server finds that the mail data could be successfully delivered to some, but not all, of the recipients. In such cases, the response to the DATA command MUST be an OK reply. However, the SMTP server MUST compose and send an "undeliverable mail" notification message to the originator of the message.

A single notification listing all of the failed recipients or separate notification messages MUST be sent for each failed recipient. For economy of processing by the sender, the former is preferred when possible. All undeliverable mail notification messages are sent using the MAIL command (even if they result from processing the obsolete SEND, SOML, or SAML commands) and use a null return path as discussed in section 3.7.

The time stamp line and the return path line are formally defined as follows:

Return-path-line = "Return-Path:" FWS Reverse-path <CRLF>

Time-stamp-line = "Received:" FWS Stamp <CRLF>

Stamp = From-domain By-domain Opt-info ";" FWS date-time; where "date-time" is as defined in [32]; but the "obs-" forms, especially two-digit; years, are prohibited in SMTP and MUST NOT be used.

From-domain = "FROM" FWS Extended-Domain CFWS

By-domain = "BY" FWS Extended-Domain CFWS

*Extended-Domain = Domain /
(Domain FWS "(" TCP-info ")") /
(Address-literal FWS "(" TCP-info ")")
TCP-info = Address-literal / (Domain FWS Address-literal)
; Information derived by server from TCP connection*

***; not client EHLO.
Opt-info = [Via] [With] [ID] [For]***

***Via = "VIA" FWS Link CFWS
With = "WITH" FWS Protocol CFWS
ID = "ID" FWS String / msg-id CFWS
For = "FOR" FWS 1*(Path / Mailbox) CFWS
Link = "TCP" / Addtl-Link
Addtl-Link = Atom
; Additional standard names for links are registered with the
; Internet Assigned Numbers Authority (IANA). "Via" is
; primarily of value with non-Internet transports. SMTP
; servers SHOULD NOT use unregistered names.
Protocol = "ESMTP" / "SMTP" / Attdl-Protocol
Attdl-Protocol = Atom
; Additional standard names for protocols are registered with the
; Internet Assigned Numbers Authority (IANA). SMTP servers
; SHOULD NOT use unregistered names.***

Sophisticated techniques now uses such fake received headers in spam detection as there are often basic format errors or data construction errors present in such headers.

Other detectable spamming techniques included using detectable templates and the complexity of new spamware became difficult for spamware-users to use, leading to simple errors, often detectable.

Enter network hash-based techniques.

- External spam database checks:
 - These include Collaborative, usually hash-based spam identification databases that compare fingerprints for received messages against shared lists of previously seen messages.
 - Spammers began adding random text to message bodies to prevent hash signatures from matching
 - When used in base64-encoded email, otherwise detected patterns no longer match
 - However, the "random" text turned out not to always be truly random and thus detectable with more sophisticated pattern matching in X-Mailer headers and Message IDs.
 - Also "random" text was often based on time-based values, again detected by more sophisticated code
 - Further sophistication, detection of substitution and rotation ciphers.
- Artificial Intelligence based learning email traffic that is continuously updated on the fly.

This technology is immune to many traditional anti-spam filter attacks used by spammers, however, random word garbage or text made up of true language dictionary words in fonts or formatted so they are unobvious or practically invisible to the user was used to attack these methods.

However, garbage text and intelligent analysis of "true language" is detectable and this attack was then largely nullified. Additionally, this technique only a portion of the overall system, used in scoring by TMD.

The resource load from a system standpoint of the above operations is extensive, hence TMD is optimized for the required memory, CPU and general infrastructure requirements to handle the processing described above.

General purpose mail servers are not optimized for such and indeed often buckle when exposed to the flood of traffic currently present on the Internet at large.

Many static solutions only mechanically use one or at best a few methods of spam analysis (such as RBL or other network-based checks, and, even worse, simply attempt to return messages to the sender based on these limited and dubious detection methods. These commonly known mail bounces are never delivered to actual spammers thus wasting system resources in generating them, and only arrived in user mailboxes that sent legitimate mail, now left with a broken communication channel. These users normally have no idea how to resolve the situation, and indeed, in the case of a sales or information communication, never bother making the attempt, but instead contact a competitor.

TMD utilizes over 12 separate and distinct spam analysis/malware/virus protection actions to determine a message's overall spam probability. Returning the message to sender (an ultimately useless and bandwidth/resource wasting exercise) is not attempted. Unique to TMD is its combination of multiple solutions utilizing industry standard open source software such as hardened server applications in production use throughout the Internet for decades; customized by TMD technical staff for one purpose: deliver only legitimate mail. To accomplish this, email is categorized for various actions by hundreds of tests returning thousands of results:

- Deletion
- Silent blocking
- Notified attachment blocking (in the case of disallowed but not necessarily malicious attachments)
- Notified spam blocking (for messages that are potentially spam) so that users can determine whether the message is legitimate and also can easily white/black list the sender.

Mail administrators have access to all mail for their domain, whitelist and blacklist, release functionality. Many organizations have found that after implementation, management overhead is extremely low as users can manage their own email traffic. Mobile users, often plagued by large quantities of spam, are particularly benefited.

CONCLUSION

Total Mail Defense provides an innovative approach to email system protection. Vast bandwidth and resource savings can be very easily had with no change to current corporate mail configuration, current mail system location and user accounts. Companies with on-site (in-house) mail servers particularly benefit from the huge reduction in risk to their IT infrastructure.

Total Mail Defense can be likened to an email firewall and like true firewalls, where once only an option or an experiment for techies, has become an absolute necessity.

***Ron Edison - Founder, Chairman & CTO
Internet Defense Technologies***