

Using TLS in Applications
Internet-Draft
Intended status: Standards Track
Expires: December 18, 2018

D. Margolis
M. Risher
Google, Inc
B. Ramakrishnan
Yahoo!, Inc
A. Brotman
Comcast, Inc
J. Jones
Microsoft, Inc
June 16, 2018

SMTP MTA Strict Transport Security (MTA-STS)
draft-ietf-uta-mta-sts-21

Abstract

SMTP Mail Transfer Agent Strict Transport Security (MTA-STS) is a mechanism enabling mail service providers to declare their ability to receive Transport Layer Security (TLS) secure SMTP connections, and to specify whether sending SMTP servers should refuse to deliver to MX hosts that do not offer TLS with a trusted server certificate.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on December 18, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology	3
2. Related Technologies	4
3. Policy Discovery	4
3.1. MTA-STS TXT Records	4
3.2. MTA-STS Policies	6
3.3. HTTPS Policy Fetching	9
3.4. Policy Selection for Smart Hosts and Subdomains	10
4. Policy Validation	10
4.1. MX Host Validation	11
4.2. Recipient MTA Certificate Validation	11
5. Policy Application	11
5.1. Policy Application Control Flow	12
6. Reporting Failures	12
7. Interoperability Considerations	13
7.1. SNI Support	13
7.2. Minimum TLS Version Support	13
8. Operational Considerations	14
8.1. Policy Updates	14
8.2. Policy Delegation	14
8.3. Removing MTA-STS	15
8.4. Preserving MX Candidate Traversal	16
9. IANA Considerations	16
9.1. Well-Known URIs Registry	16
9.2. MTA-STS TXT Record Fields	16
9.3. MTA-STS Policy Fields	17
10. Security Considerations	17
10.1. Obtaining a Signed Certificate	17
10.2. Preventing Policy Discovery	18
10.3. Denial of Service	18
10.4. Weak Policy Constraints	19
10.5. Compromise of the Web PKI System	19
11. Contributors	20
12. References	20
12.1. Normative References	20
12.2. Informative References	22
Appendix A. MTA-STS example record & policy	23
Appendix B. Message delivery pseudocode	23
Authors' Addresses	26

1. Introduction

The STARTTLS extension to SMTP [RFC3207] allows SMTP clients and hosts to negotiate the use of a TLS channel for encrypted mail transmission.

While this opportunistic encryption protocol by itself provides a high barrier against passive man-in-the-middle traffic interception, any attacker who can delete parts of the SMTP session (such as the "250 STARTTLS" response) or who can redirect the entire SMTP session (perhaps by overwriting the resolved MX record of the delivery domain) can perform downgrade or interception attacks.

This document defines a mechanism for recipient domains to publish policies, via a combination of DNS and HTTPS, specifying:

- o whether MTAs sending mail to this domain can expect PKIX-authenticated TLS support
- o what a conforming client should do with messages when TLS cannot be successfully negotiated

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BCP 14] [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

We also define the following terms for further use in this document:

- o MTA-STS Policy: A commitment by the Policy Domain to support PKIX [RFC5280] authenticated TLS for the specified MX hosts.
- o Policy Domain: The domain for which an MTA-STS Policy is defined. This is the next-hop domain; when sending mail to "alice@example.com" this would ordinarily be "example.com", but this may be overridden by explicit routing rules (as described in Section 3.4, "Policy Selection for Smart Hosts and Subdomains").
- o Policy Host: The HTTPS host which serves the MTA-STS Policy for a Policy Domain. Rules for constructing the hostname are described in Section 3.2, "MTA-STS Policies".
- o Sender: The SMTP Mail Transfer Agent sending an email message.

- o ABNF: Augmented Backus-Naur Form, a syntax for formally specifying syntax, defined in [RFC5234] and [RFC7405].

2. Related Technologies

The DANE TLSA record [RFC7672] is similar, in that DANE is also designed to upgrade unauthenticated encryption or plaintext transmission into authenticated, downgrade-resistant encrypted transmission. DANE requires DNSSEC [RFC4033] for authentication; the mechanism described here instead relies on certificate authorities (CAs) and does not require DNSSEC, at a cost of risking malicious downgrades. For a thorough discussion of this trade-off, see Section 10, "Security Considerations".

In addition, MTA-STS provides an optional testing-only mode, enabling soft deployments to detect policy failures; partial deployments can be achieved in DANE by deploying TLSA records only for some of a domain's MXs, but such a mechanism is not possible for the per-domain policies used by MTA-STS.

The primary motivation of MTA-STS is to provide a mechanism for domains to ensure transport security even when deploying DNSSEC is undesirable or impractical. However, MTA-STS is designed not to interfere with DANE deployments when the two overlap; in particular, senders who implement MTA-STS validation MUST NOT allow a "valid" or "testing"-only MTA-STS validation to override a failing DANE validation.

3. Policy Discovery

MTA-STS policies are distributed via HTTPS from a "well-known" [RFC5785] path served within the Policy Domain, and their presence and current version are indicated by a TXT record at the Policy Domain. These TXT records additionally contain a policy "id" field, allowing sending MTAs to check the currency of a cached policy without performing an HTTPS request.

To discover if a recipient domain implements MTA-STS, a sender need only resolve a single TXT record. To see if an updated policy is available for a domain for which the sender has a previously cached policy, the sender need only check the TXT record's version "id" against the cached value.

3.1. MTA-STS TXT Records

The MTA-STS TXT record is a TXT record with the name "_mta-sts" at the Policy Domain. For the domain "example.com", this record would

be "_mta-sts.example.com". MTA-STS TXT records MUST be US-ASCII, semicolon-separated key/value pairs containing the following fields:

- o "v": (plain-text, required). Currently only "STSV1" is supported.
- o "id": (plain-text, required). A short string used to track policy updates. This string MUST uniquely identify a given instance of a policy, such that senders can determine when the policy has been updated by comparing to the "id" of a previously seen policy. There is no implied ordering of "id" fields between revisions.

An example TXT record is as below:

```
_mta-sts.example.com. IN TXT "v=STSV1; id=20160831085700Z;"
```

The formal definition of the "_mta-sts" TXT record, defined using ABNF ([RFC7405]), is as follows:

```
sts-text-record = sts-version 1*(sts-field-delim sts-field)
                  [sts-field-delim]

sts-field        = sts-id /                               ; Note that sts-id record
                  sts-extension                             ; is required.

sts-field-delim  = *WSP ";" *WSP

sts-version      = %s"v=STSV1"

sts-id           = %s"id=" 1*32(ALPHA / DIGIT)             ; id=...

sts-extension    = sts-ext-name "=" sts-ext-value        ; name=value

sts-ext-name     = (ALPHA / DIGIT)
                  *31(ALPHA / DIGIT / "_" / "-" / ".")

sts-ext-value    = 1*(%x21-3A / %x3C / %x3E-7E)
                  ; chars excluding "=", ";", SP, and CTLs
```

The TXT record MUST begin with sts-version field, and the order of other fields is not significant. If multiple TXT records for "_mta-sts" are returned by the resolver, records which do not begin with "v=STSV1;" are discarded. If the number of resulting records is not one, or if the resulting record is syntactically invalid, senders MUST assume the recipient domain does not have an available MTA-STS policy and skip the remaining steps of policy discovery. (Note that absence of a usable TXT record is not by itself sufficient to remove a sender's previously cached policy for the Policy Domain, as discussed in Section 5.1, "Policy Application Control Flow".) If the

resulting TXT record contains multiple strings, then the record MUST be treated as if those strings are concatenated together without adding spaces.

3.2. MTA-STS Policies

The policy itself is a set of key/value pairs (similar to [RFC5322] header fields) served via the HTTPS GET method from the fixed [RFC5785] "well-known" path of ".well-known/mta-sts.txt" served by the Policy Host. The Policy Host DNS name is constructed by prepending "mta-sts" to the Policy Domain.

Thus for a Policy Domain of "example.com" the full URL is "https://mta-sts.example.com/.well-known/mta-sts.txt".

When fetching a policy, senders SHOULD validate that the media type is "text/plain" to guard against cases where web servers allow untrusted users to host non-text content (typically, HTML or images) at a user-defined path. All parameters other than charset=utf-8 or charset=us-ascii are ignored. Additional "Content-Type" parameters are also ignored.

This resource contains the following CRLF-separated key/value pairs:

- o "version": Currently only "STSV1" is supported.
- o "mode": One of "enforce", "testing", or "none", indicating the expected behavior of a sending MTA in the case of a policy validation failure. See Section 5, "Policy Application." for more details about the three modes.
- o "max_age": Max lifetime of the policy (plain-text non-negative integer seconds, maximum value of 31557600). Well-behaved clients SHOULD cache a policy for up to this value from last policy fetch time. To mitigate the risks of attacks at policy refresh time, it is expected that this value typically be in the range of weeks or greater.
- o "mx": Allowed MX patterns. One or more patterns matching allowed MX hosts for the Policy Domain. As an example,

```
mx: mail.example.com <CRLF>
mx: *.example.net
```

indicates that mail for this domain might be handled by MX "mail.example.com" or any MX at "example.net". Valid patterns can be either fully specified names ("example.com") or suffixes prefixed by a wildcard ("*.example.net"). If a policy specifies more than one

MX, each MX MUST have its own "mx:" key, and each MX key/value pair MUST be on its own line in the policy file. In the case of Internationalized Domain Names ([RFC5891]), the "mx" value MUST specify the Punycode-encoded A-label [RFC3492] to match against, and not the Unicode-encoded U-label. The full semantics of certificate validation (including the use of wildcard patterns) are described in Section 4.1, "MX Host Validation."

An example policy is as below:

```
version: STSv1
mode: enforce
mx: mail.example.com
mx: *.example.net
mx: backupmx.example.com
max_age: 604800
```

The formal definition of the policy resource, defined using [RFC7405], is as follows:

```
sts-policy-record      = sts-policy-field *WSP
                        *(sts-policy-term sts-policy-field *WSP)
                        [sts-policy-term]

sts-policy-field       = sts-policy-version /           ; required once
                        sts-policy-mode /               ; required once
                        sts-policy-max-age /            ; required once

                        sts-policy-term /
                        ; required at least once, except when
                        ; mode is "none"

                        sts-policy-extension            ; other fields

sts-policy-field-delim = ":" *WSP

sts-policy-version     = sts-policy-version-field sts-policy-field-delim
                        sts-policy-version-value

sts-policy-version-field = %s"version"

sts-policy-version-value = %s"STSv1"

sts-policy-mode        = sts-policy-mode-field sts-policy-field-delim
                        sts-policy-mode-value

sts-policy-mode-field  = %s"mode"
```

sts-policy-mode-value = %s"testing" / %s"enforce" / %s"none"

sts-policy-mx = sts-policy-mx-field sts-policy-field-delim
sts-policy-mx-value

sts-policy-mx-field = %s"mx"

sts-policy-mx-value = ["."] Domain

sts-policy-mx-label = sts-policy-alphanum /
sts-policy-alphanum *(sts-policy-alphanum / "-")
sts-policy-alphanum

sts-policy-mx-toplabel = ALPHA / ALPHA *(sts-policy-alphanum / "-")
sts-policy-alphanum

sts-policy-max-age = sts-policy-max-age-field sts-policy-field-delim
sts-policy-max-age-value

sts-policy-max-age-field = %s"max_age"

sts-policy-max-age-value = 1*10(DIGIT)

sts-policy-extension = sts-policy-ext-name ; additional
sts-policy-field-delim ; extension
sts-policy-ext-value ; fields

sts-policy-ext-name = (sts-policy-alphanum)
*31(sta-policy-alphanum / "_" / "-" / ".")

sts-policy-term = LF / CRLF

sts-policy-ext-value = sts-policy-vchar
[*(%x20 / sts-policy-vchar)
sts-policy-vchar]
; chars, including UTF-8 [!RFC3629],
; excluding CTLs and no
; leading/trailing spaces

sts-policy-alphanum = ALPHA / DIGIT

sts-policy-vchar = %x21-7E / UTF8-2 / UTF8-3 / UTF8-4

UTF8-2 = <Defined in Section 4 of [!RFC3629]>

UTF8-3 = <Defined in Section 4 of [!RFC3629]>

UTF8-4 = <Defined in Section 4 of [!RFC3629]>

Domain = <see RFC 5321 4.1.2>

Parsers MUST accept TXT records and policy files which are syntactically valid (i.e., valid key/value pairs separated by semi-colons for TXT records), possibly containing additional key/value pairs not specified in this document, in which case unknown fields SHALL be ignored. If any non-repeated field--i.e., all fields excepting "mx"--is duplicated, all entries except for the first SHALL be ignored.

3.3. HTTPS Policy Fetching

Policy bodies are, as described above, retrieved by sending MTAs via HTTPS [RFC2818]. During the TLS handshake initiated to fetch a new or updated policy from the Policy Host, the Policy Host HTTPS server MUST present a X.509 certificate which is valid for the "mta-sts" DNS-ID ([RFC6125]) (e.g., "mta-sts.example.com") as described below, chain to a root CA that is trusted by the sending MTA, and be non-expired. It is expected that sending MTAs use a set of trusted CAs similar to those in widely deployed Web browsers and operating systems. See [RFC5280] for more details about certificate verification.

The certificate is valid for the Policy Host (i.e., "mta-sts" prepended to the Policy Domain) with respect to the rules described in [RFC6125], with the following application-specific considerations:

- o Matching is performed only against the DNS-ID identifiers.
- o DNS domain names in server certificates MAY contain the wildcard character '*' as the complete left-most label within the identifier.

The certificate MAY be checked for revocation via the Online Certificate Status Protocol (OCSP) [RFC6960], certificate revocation lists (CRLs), or some other mechanism.

Policies fetched via HTTPS are only valid if the HTTP response code is 200 (OK). HTTP 3xx redirects MUST NOT be followed, and HTTP caching (as specified in [RFC7234]) MUST NOT be used.

Senders may wish to rate-limit the frequency of attempts to fetch the HTTPS endpoint even if a valid TXT record for the recipient domain exists. In the case that the HTTPS GET fails, implementers SHOULD limit further attempts to a period of five minutes or longer per version ID, to avoid overwhelming resource-constrained recipients with cascading failures.

Senders MAY impose a timeout on the HTTPS GET and/or a limit on the maximum size of the response body to avoid long delays or resource exhaustion during attempted policy updates. A suggested timeout is one minute, and a suggested maximum policy size 64 kilobytes; policy hosts SHOULD respond to requests with a complete policy body within that timeout and size limit.

If a valid TXT record is found but no policy can be fetched via HTTPS (for any reason), and there is no valid (non-expired) previously-cached policy, senders MUST continue with delivery as though the domain has not implemented MTA-STS.

Conversely, if no "live" policy can be discovered via DNS or fetched via HTTPS, but a valid (non-expired) policy exists in the sender's cache, the sender MUST apply that cached policy.

Finally, to mitigate the risk of persistent interference with policy refresh, as discussed in-depth in Section 10, MTAs SHOULD proactively refresh cached policies before they expire; a suggested refresh frequency is once per day. To enable administrators to discover problems with policy refresh, MTAs SHOULD alert administrators (through the use of logs or similar) when such attempts fail, unless the cached policy mode is "none".

3.4. Policy Selection for Smart Hosts and Subdomains

When sending mail via a "smart host"--an administratively configured intermediate SMTP relay, which is different from the message recipient's server as determined from DNS --compliant senders MUST treat the smart host domain as the policy domain for the purposes of policy discovery and application. This specification does not provide a means of associating policies with addresses that employ Address Literals [RFC5321].

When sending mail to a mailbox at a subdomain, compliant senders MUST NOT attempt to fetch a policy from the parent zone. Thus for mail sent to "user@mail.example.com", the policy can be fetched only from "mail.example.com", not "example.com".

4. Policy Validation

When sending to an MX at a domain for which the sender has a valid and non-expired MTA-STS policy, a sending MTA honoring MTA-STS MUST check whether:

1. At least one of the policy's "mx" patterns matches the selected MX host, as described in Section 4.1, "MX Host Validation".

2. The recipient mail server supports STARTTLS and offers a PKIX-based TLS certificate, during TLS handshake, which is valid for that host, as described in Section 4.2, "Recipient MTA Certificate Validation".

When these conditions are not met, a policy is said to fail to validate. This section does not dictate the behavior of sending MTAs when the above conditions are not met; see Section 5, "Policy Application" for a description of sending MTA behavior when policy validation fails.

4.1. MX Host Validation

A receiving candidate MX host is valid according to an applied MTA-STS policy if the MX record name matches one or more of the "mx" fields in the applied policy. Matching is identical to the rules given in [RFC6125], with restriction that the wildcard character "*" may only be used to match the entire left-most label in the presented identifier. Thus the mx pattern "*.example.com" matches "mail.example.com" but not "example.com" or "foo.bar.example.com".

4.2. Recipient MTA Certificate Validation

The certificate presented by the receiving MTA MUST not be expired, and MUST chain to a root CA that is trusted by the sending MTA. The certificate MUST have a subject alternative name (SAN, [RFC5280]) with a DNS-ID ([RFC6125]) matching the host name, per the rules given in [RFC6125]. The MX's certificate MAY also be checked for revocation via OCSP [RFC6960], CRLs [RFC6818], or some other mechanism.

5. Policy Application

When sending to an MX at a domain for which the sender has a valid, non-expired MTA-STS policy, a sending MTA honoring MTA-STS applies the result of a policy validation failure one of two ways, depending on the value of the policy "mode" field:

1. "enforce": In this mode, sending MTAs MUST NOT deliver the message to hosts which fail MX matching or certificate validation, or do not support STARTTLS.
2. "testing": In this mode, sending MTAs which also implement the TLSRPT specification [I-D.ietf-uta-smtp-tlsrpt] merely send a report indicating policy application failures (so long as TLSRPT is also implemented by the recipient domain).

3. "none": In this mode, sending MTAs should treat the policy domain as though it does not have any active policy; see Section 8.3, "Removing MTA-STS", for use of this mode value.

When a message fails to deliver due to an "enforce" policy, a compliant MTA MUST NOT permanently fail to deliver messages before checking, via DNS, for the presence of an updated policy at the Policy Domain. (In all cases, MTAs SHOULD treat such failures as transient errors and retry delivery later.) This allows implementing domains to update long-lived policies on the fly.

5.1. Policy Application Control Flow

An example control flow for a compliant sender consists of the following steps:

1. Check for a cached policy whose time-since-fetch has not exceeded its "max_age". If none exists, attempt to fetch a new policy (perhaps asynchronously, so as not to block message delivery). Optionally, sending MTAs may unconditionally check for a new policy at this step.
2. For each candidate MX, in order of MX priority, attempt to deliver the message. If a policy is present with an "enforce" mode, when attempting to deliver to each candidate MX, ensure STARTTLS support and host identity validity as described in Section 4, "Policy Validation". If a candidate fails validation, continue to the next candidate (if there is one).
3. A message delivery MUST NOT be permanently failed until the sender has first checked for the presence of a new policy (as indicated by the "id" field in the "_mta-sts" TXT record). If a new policy is not found, existing rules for the case of temporary message delivery failures apply (as discussed in [RFC5321] section 4.5.4.1).

6. Reporting Failures

MTA-STS is intended to be used along with TLSRPT [I-D.ietf-uta-smtp-tlsrpt] in order to ensure implementing domains can detect cases of both benign and malicious failures, and to ensure that failures that indicate an active attack are discoverable. As such, senders who also implement TLSRPT SHOULD treat the following events as reportable failures:

- o HTTPS policy fetch failures when a valid TXT record is present.

- o Policy fetch failures of any kind when a valid policy exists in the policy cache, except if that policy's mode is "none".
- o Delivery attempts in which a contacted MX does not support STARTTLS or does not present a certificate which validates according to the applied policy, except if that policy's mode is "none".

7. Interoperability Considerations

7.1. SNI Support

To ensure that the server sends the right certificate chain, the SMTP client **MUST** have support for the TLS SNI extension [RFC6066]. When connecting to a HTTP server to retrieve the MTA-STS policy, the SNI extension **MUST** contain the name of the policy host (e.g., "mta-sts.example.com"). When connecting to an SMTP server, the SNI extension **MUST** contain the MX hostname.

HTTP servers used to deliver MTA-STS policies **MAY** rely on SNI to determine which certificate chain to present to the client. HTTP servers **MUST** respond with a certificate chain that matches the policy hostname or abort the TLS handshake if unable to do so. Clients that do not send SNI information may not see the expected certificate chain.

SMTP servers **MAY** rely on SNI to determine which certificate chain to present to the client. However servers that have one identity and a single matching certificate do not require SNI support. Servers **MUST NOT** enforce the use of SNI by clients, as the client may be using unauthenticated opportunistic TLS and may not expect any particular certificate from the server. If the client sends no SNI extension or sends an SNI extension for an unsupported server name, the server **MUST** simply send a fallback certificate chain of its choice. The reason for not enforcing strict matching of the requested SNI hostname is that MTA-STS TLS clients may be typically willing to accept multiple server names but can only send one name in the SNI extension. The server's fallback certificate may match a different name that is acceptable to the client, e.g., the original next-hop domain.

7.2. Minimum TLS Version Support

MTAs supporting MTA-STS **MUST** have support for TLS version 1.2 [RFC5246] or higher. The general TLS usage guidance in [RFC7525] **SHOULD** be followed.

8. Operational Considerations

8.1. Policy Updates

Updating the policy requires that the owner make changes in two places: the "_mta-sts" TXT record in the Policy Domain's DNS zone and at the corresponding HTTPS endpoint. As a result, recipients should expect a policy will continue to be used by senders until both the HTTPS and TXT endpoints are updated and the TXT record's TTL has passed.

In other words, a sender who is unable to successfully deliver a message while applying a cache of the recipient's now-outdated policy may be unable to discover that a new policy exists until the DNS TTL has passed. Recipients SHOULD therefore ensure that old policies continue to work for message delivery during this period of time, or risk message delays.

Recipients SHOULD also update the HTTPS policy body before updating the TXT record; this ordering avoids the risk that senders, seeing a new TXT record, mistakenly cache the old policy from HTTPS.

8.2. Policy Delegation

Domain owners commonly delegate SMTP hosting to a different organization, such as an ISP or a Web host. In such a case, they may wish to also delegate the MTA-STS policy to the same organization which can be accomplished with two changes.

First, the Policy Domain must point the "_mta-sts" record, via CNAME, to the "_mta-sts" record maintained by the hosting organization. This allows the hosting organization to control update signaling.

Second, the Policy Domain must point the "well-known" policy location to the hosting organization. This can be done either by setting the "mta-sts" record to an IP address or CNAME specified by the hosting organization and by giving the hosting organization a TLS certificate which is valid for that host, or by setting up a "reverse proxy" (also known as a "gateway") server that serves as the Policy Domain's policy the policy currently served by the hosting organization.

For example, given a user domain "user.example" hosted by a mail provider "provider.example", the following configuration would allow policy delegation:

DNS:

```
_mta-sts.user.example. IN CNAME _mta-sts.provider.example.
```

Policy:

```
> GET /.well-known/mta-sts.txt Host: mta-sts.user.example
< HTTP/1.1 200 OK    # Response proxies content from
                     # https://mta-sts.provider.example
```

Note that in all such cases, the policy endpoint ("https://mta-sts.user.example/.well-known/mta-sts.txt" in this example) must still present a certificate valid for the Policy Host ("mta-sts.user.example"), and not for that host at the provider's domain ("mta-sts.provider.example").

Note that while sending MTAs MUST NOT use HTTP caching when fetching policies via HTTPS, such caching may nonetheless be useful to a reverse proxy configured as described in this section. An HTTPS policy endpoint expecting to be proxied for multiple hosted domains--as with a large mail hosting provider or similar--may wish to indicate an HTTP Cache-Control "max-age" response directive (as specified in [RFC7234]) of 60 seconds as a reasonable value to save reverse proxies an unnecessarily high-rate of proxied policy fetching.

8.3. Removing MTA-STS

In order to facilitate clean opt-out of MTA-STS by implementing policy domains, and to distinguish clearly between failures which indicate attacks and those which indicate such opt-outs, MTA-STS implements the "none" mode, which allows validated policies to indicate authoritatively that the policy domain wishes to no longer implement MTA-STS and may, in the future, remove the MTA-STS TXT and policy endpoints entirely.

A suggested workflow to implement such an opt out is as follows:

1. Publish a new policy with "mode" equal to "none" and a small "max_age" (e.g., one day).
2. Publish a new TXT record to trigger fetching of the new policy.
3. When all previously served policies have expired--normally this is the time the previously published policy was last served plus that policy's "max_age", but note that older policies may have been served with a greater "max_age", allowing overlapping policy caches--safely remove the TXT record and HTTPS endpoint.

8.4. Preserving MX Candidate Traversal

Implementors of send-time MTA-STS validation in mail transfer agents should take note of the risks of modifying the logic of traversing MX candidate lists. Because an MTA-STS policy can be used to prefilter invalid MX candidates from the MX candidate list, it is tempting to implement a "two-pass" model, where MX candidates are first filtered for possible validity according to the MTA-STS policy, and then the remaining candidates attempted in order as without an MTA-STS policy. This may lead to incorrect implementations, such as message loops; implementors are instead recommended to traverse the MX candidate list as usual, and treat invalid candidates as though they were unreachable (i.e., as though there were some transient error when trying to deliver to that candidate).

One consequence of validating MX hosts in order of ordinary candidate traversal is that, in the event that a higher-priority MX is MTA-STS valid and a lower-priority MX is not, senders may never encounter the lower-priority MX, leading to a risk that policy misconfigurations that apply only to "backup" MXes may only be discovered in the case of primary MX failure.

9. IANA Considerations

9.1. Well-Known URIs Registry

A new "well-known" URI as described in Section 3 will be registered in the Well-Known URIs registry as described below:

URI Suffix: mta-sts.txt Change Controller: IETF

9.2. MTA-STS TXT Record Fields

IANA is requested to create a new registry titled "MTA-STS TXT Record Fields". The initial entries in the registry are:

Field Name	Description	Reference
v	Record version	Section 3.1 of RFC XXX
id	Policy instance ID	Section 3.1 of RFC XXX

New fields are added to this registry using IANA's "Expert Review" policy.

9.3. MTA-STS Policy Fields

IANA is requested to create a new registry titled "MTA-STS Policy Fields". The initial entries in the registry are:

Field Name	Description	Reference
version	Policy version	Section 3.2 of RFC XXX
mode	Enforcement behavior	Section 3.2 of RFC XXX
max_age	Policy lifetime	Section 3.2 of RFC XXX
mx	MX identities	Section 3.2 of RFC XXX

New fields are added to this registry using IANA's "Expert Review" policy.

10. Security Considerations

SMTP MTA Strict Transport Security attempts to protect against an active attacker trying to intercept or tamper with mail between hosts that support STARTTLS. There are two classes of attacks considered:

- o Foiling TLS negotiation, for example by deleting the "250 STARTTLS" response from a server or altering TLS session negotiation. This would result in the SMTP session occurring over plaintext, despite both parties supporting TLS.
- o Impersonating the destination mail server, whereby the sender might deliver the message to an impostor, who could then monitor and/or modify messages despite opportunistic TLS. This impersonation could be accomplished by spoofing the DNS MX record for the recipient domain, or by redirecting client connections intended for the legitimate recipient server (for example, by altering BGP routing tables).

MTA-STS can thwart such attacks only if the sender is able to previously obtain and cache a policy for the recipient domain, and only if the attacker is unable to obtain a valid certificate that complies with that policy. Below, we consider specific attacks on this model.

10.1. Obtaining a Signed Certificate

SMTP MTA-STS relies on certificate validation via PKIX based TLS identity checking [RFC6125]. Attackers who are able to obtain a valid certificate for the targeted recipient mail service (e.g., by

compromising a certificate authority) are thus able to circumvent STS authentication.

10.2. Preventing Policy Discovery

Since MTA-STS uses DNS TXT records for policy discovery, an attacker who is able to block DNS responses can suppress the discovery of an MTA-STS Policy, making the Policy Domain appear not to have an MTA-STS Policy. The sender policy cache is designed to resist this attack by decreasing the frequency of policy discovery and thus reducing the window of vulnerability; it is nonetheless a risk that attackers who can predict or induce policy discovery--for example, by inducing a sending domain to send mail to a never-before-contacted recipient while carrying out a man-in-the-middle attack--may be able to foil policy discovery and effectively downgrade the security of the message delivery.

Since this attack depends upon intercepting initial policy discovery, implementers SHOULD prefer policy "max_age" values to be as long as is practical.

Because this attack is also possible upon refresh of a cached policy, implementors SHOULD NOT wait until a cached policy has expired before checking for an update; if senders attempt to refresh the cache regularly (for example, by fetching currently live policy in a background task that runs daily or weekly, regardless of the state of the "_mta_sts" TXT record, and updating their cache's "max age" accordingly), an attacker would have to foil policy discovery consistently over the lifetime of a cached policy to prevent a successful refresh.

Additionally, MTAs SHOULD alert administrators to repeated policy refresh failures long before cached policies expire (through warning logs or similar applicable mechanisms), allowing administrators to detect such a persistent attack on policy refresh. (However, they should not implement such alerts if the cached policy has a "none" mode, to allow clean MTA-STS removal, as described in Section 8.3.)

Resistance to downgrade attacks of this nature--due to the ability to authoritatively determine "lack of a record" even for non-participating recipients--is a feature of DANE, due to its use of DNSSEC for policy discovery.

10.3. Denial of Service

We additionally consider the Denial of Service risk posed by an attacker who can modify the DNS records for a recipient domain. Absent MTA-STS, such an attacker can cause a sending MTA to cache

invalid MX records, but only for however long the sending resolver caches those records. With MTA-STS, the attacker can additionally advertise a new, long-"max_age" MTA-STS policy with "mx" constraints that validate the malicious MX record, causing senders to cache the policy and refuse to deliver messages once the victim has resecured the MX records.

This attack is mitigated in part by the ability of a victim domain to (at any time) publish a new policy updating the cached, malicious policy, though this does require the victim domain to both obtain a valid CA-signed certificate and to understand and properly configure MTA-STS.

Similarly, we consider the possibility of domains that deliberately allow untrusted users to serve untrusted content on user-specified subdomains. In some cases (e.g., the service Tumblr.com) this takes the form of providing HTTPS hosting of user-registered subdomains; in other cases (e.g. dynamic DNS providers) this takes the form of allowing untrusted users to register custom DNS records at the provider's domain.

In these cases, there is a risk that untrusted users would be able to serve custom content at the "mta-sts" host, including serving an illegitimate MTA-STS policy. We believe this attack is rendered more difficult by the need for the attacker to also serve the "_mta-sts" TXT record on the same domain--something not, to our knowledge, widely provided to untrusted users. This attack is additionally mitigated by the aforementioned ability for a victim domain to update an invalid policy at any future date.

10.4. Weak Policy Constraints

Even if an attacker cannot modify a served policy, the potential exists for configurations that allow attackers on the same domain to receive mail for that domain. For example, an easy configuration option when authoring an MTA-STS Policy for "example.com" is to set the "mx" equal to "*.example.com"; recipient domains must consider in this case the risk that any user possessing a valid hostname and CA-signed certificate (for example, "dhcp-123.example.com") will, from the perspective of MTA-STS Policy validation, be a valid MX host for that domain.

10.5. Compromise of the Web PKI System

A host of risks apply to the PKI system used for certificate authentication, both of the "mta-sts" HTTPS host's certificate and the SMTP servers' certificates. These risks are broadly applicable

within the Web PKI ecosystem and are not specific to MTA-STS; nonetheless, they deserve some consideration in this context.

Broadly speaking, attackers may compromise the system by obtaining certificates under fraudulent circumstances (i.e., by impersonating the legitimate owner of the victim domain), by compromising a Certificate Authority or Delegate Authority's private keys, by obtaining a legitimate certificate issued to the victim domain, and similar.

One approach commonly employed by Web browsers to help mitigate against some of these attacks is to allow for revocation of compromised or fraudulent certificates via OCSP [RFC6960] or CRLs [RFC6818]. Such mechanisms themselves represent tradeoffs and are not universally implemented; we nonetheless recommend implementors of MTA-STS to implement revocation mechanisms which are most applicable to their implementations.

11. Contributors

Wei Chuang Google, Inc weihaw@google.com

Viktor Dukhovni ietf-dane@dukhovni.de

Markus Laber 1&1 Mail & Media Development & Technology GmbH
markus.laber@lund1.de

Nicolas Lidzborski Google, Inc nlidz@google.com

Brandon Long Google, Inc blong@google.com

Franck Martin LinkedIn, Inc fmartin@linkedin.com

Klaus Umbach 1&1 Mail & Media Development & Technology GmbH
klaus.umbach@lund1.de

12. References

12.1. Normative References

- [I-D.ietf-uta-smtp-tlsrpt]
Margolis, D., Brotman, A., Ramakrishnan, B., Jones, J.,
and M. Risher, "SMTP TLS Reporting", draft-ietf-uta-smtp-
tlsrpt-22 (work in progress), May 2018.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<https://www.rfc-editor.org/info/rfc2818>>.
- [RFC3207] Hoffman, P., "SMTP Service Extension for Secure SMTP over Transport Layer Security", RFC 3207, DOI 10.17487/RFC3207, February 2002, <<https://www.rfc-editor.org/info/rfc3207>>.
- [RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, DOI 10.17487/RFC3492, March 2003, <<https://www.rfc-editor.org/info/rfc3492>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <<https://www.rfc-editor.org/info/rfc5321>>.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <<https://www.rfc-editor.org/info/rfc5785>>.
- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/info/rfc6066>>.

- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC7405] Kyzivat, P., "Case-Sensitive String Support in ABNF", RFC 7405, DOI 10.17487/RFC7405, December 2014, <<https://www.rfc-editor.org/info/rfc7405>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

12.2. Informative References

- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.
- [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <<https://www.rfc-editor.org/info/rfc5322>>.
- [RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <<https://www.rfc-editor.org/info/rfc5891>>.
- [RFC6818] Yee, P., "Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 6818, DOI 10.17487/RFC6818, January 2013, <<https://www.rfc-editor.org/info/rfc6818>>.
- [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <<https://www.rfc-editor.org/info/rfc6960>>.

- [RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <<https://www.rfc-editor.org/info/rfc7234>>.
- [RFC7672] Dukhovni, V. and W. Hardaker, "SMTP Security via Opportunistic DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS)", RFC 7672, DOI 10.17487/RFC7672, October 2015, <<https://www.rfc-editor.org/info/rfc7672>>.

Appendix A. MTA-STS example record & policy

The owner of "example.com" wishes to begin using MTA-STS with a policy that will solicit reports from senders without affecting how the messages are processed, in order to verify the identity of MXs that handle mail for "example.com", confirm that TLS is correctly used, and ensure that certificates presented by the recipient MX validate.

MTA-STS policy indicator TXT RR:

```
_mta-sts.example.com.  IN TXT "v=STSV1; id=20160831085700Z;"
```

MTA-STS Policy file served as the response body at "<https://mta-sts.example.com/.well-known/mta-sts.txt>":

```
version: STSV1
mode: testing
mx: mx1.example.com
mx: mx2.example.com
mx: mx.backup-example.com
max_age: 1296000
```

Appendix B. Message delivery pseudocode

Below is pseudocode demonstrating the logic of a compliant sending MTA.

While this pseudocode implementation suggests synchronous policy retrieval in the delivery path, in a working implementation that may be undesirable, and we expect some implementers to instead prefer a background fetch that does not block delivery if no cached policy is present.

```
func isEnforce(policy) {
    // Return true if the policy mode is "enforce".
```

```
}

func isNonExpired(policy) {
    // Return true if the policy is not expired.
}

func tryStartTls(connection) {
    // Attempt to open an SMTP connection with STARTTLS with the MX.
}

func certMatches(connection, host) {
    // Assume a handy function to return check if the server certificate presented
    // in "connection" is valid for "host".
}

func policyMatches(candidate, policy) {
    for mx in policy.mx {
        // Literal match.
        if mx == candidate {
            return true
        }
        // Wildcard matches only the leftmost label.
        // Wildcards must always be followed by a '.'.
        if mx[0] == '*' {
            parts = SplitN(candidate, '.', 2) // Split on the first '.'.
            if len(parts) > 1 && parts[1] == mx[2:] {
                return true
            }
        }
    }
    return false
}

func tryDeliverMail(connection, message) {
    // Attempt to deliver "message" via "connection".
}

func tryGetNewPolicy(domain) {
    // Check for an MTA-STS TXT record for "domain" in DNS, and return the
    // indicated policy.
}

func cachePolicy(domain, policy) {
    // Store "policy" as the cached policy for "domain".
}

func tryGetCachedPolicy(domain) {
    // Return a cached policy for "domain".
}
```



```
}

func reportError(error) {
    // Report an error via TLSRPT.
}

func tryMxAccordingTo(message, mx, policy) {
    connection := connect(mx)
    if !connection {
        return false // Can't connect to the MX so it's not an MTA-STS
                      // error.
    }
    secure := true
    if !policyMatches(mx, policy) {
        secure = false
        reportError(E_HOST_MISMATCH)
    } else if !tryStartTls(connection) {
        secure = false
        reportError(E_NO_VALID_TLS)
    } else if !certMatches(connection, policy) {
        secure = false
        reportError(E_CERT_MISMATCH)
    }
    if secure || !isEnforce(policy) {
        return tryDeliverMail(connection, message)
    }
    return false
}

func tryWithPolicy(message, domain, policy) {
    mxes := getMxForDomain(domain)
    for mx in mxes {
        if tryMxAccordingTo(message, mx, policy) {
            return true
        }
    }
    return false
}

func handleMessage(message) {
    domain := ... // domain part after '@' from recipient
    policy := tryGetNewPolicy(domain)
    if policy {
        cachePolicy(domain, policy)
    } else {
        policy = tryGetCachedPolicy(domain)
    }
    if policy {
```

```
    return tryWithPolicy(message, domain, policy)
  }
  // Try to deliver the message normally (i.e., without MTA-STS).
}
```

Authors' Addresses

Daniel Margolis
Google, Inc

Email: dmargolis@google.com

Mark Risher
Google, Inc

Email: risher@google.com

Binu Ramakrishnan
Yahoo!, Inc

Email: rbinu@yahoo-inc.com

Alexander Brotman
Comcast, Inc

Email: alex_brotman@comcast.com

Janet Jones
Microsoft, Inc

Email: janet.jones@microsoft.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: February 3, 2020

J. Fenton
Altmode Networks
August 2, 2019

SMTP Require TLS Option
draft-ietf-uta-smtp-require-tls-09

Abstract

The SMTP STARTTLS option, used in negotiating transport-level encryption of SMTP connections, is not as useful from a security standpoint as it might be because of its opportunistic nature; message delivery is, by default, prioritized over security. This document describes an SMTP service extension, REQUIRETLS, and message header field, TLS-Required. If the REQUIRETLS option or TLS-Required message header field is used when sending a message, it asserts a request on the part of the message sender to override the default negotiation of TLS, either by requiring that TLS be negotiated when the message is relayed, or by requesting that recipient-side policy mechanisms such as MTA-STS and DANE be ignored when relaying a message for which security is unimportant.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

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."

This Internet-Draft will expire on February 3, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	4
2. The REQUIRETLS Service Extension	4
3. The TLS-Required Header Field	5
4. REQUIRETLS Semantics	6
4.1. REQUIRETLS Receipt Requirements	6
4.2. REQUIRETLS Sender Requirements	6
4.2.1. Sending with TLS Required	6
4.2.2. Sending with TLS Optional	7
4.3. REQUIRETLS Submission	8
4.4. Delivery of REQUIRETLS messages	8
5. Non-delivery message handling	8
6. Reorigination considerations	9
7. IANA Considerations	10
8. Security Considerations	11
8.1. Passive attacks	11
8.2. Active attacks	11
8.3. Bad Actor MTAs	12
8.4. Policy Conflicts	12
9. Acknowledgements	12
10. Revision History	13
10.1. Changes since -08 Draft	13
10.2. Changes since -07 Draft	13
10.3. Changes since -06 Draft	13
10.4. Changes since -05 Draft	14
10.5. Changes since -04 Draft	14
10.6. Changes since -03 Draft	14
10.7. Changes since -02 Draft	14
10.8. Changes since -01 Draft	14
10.9. Changes since -00 Draft	15
10.10. Changes since fenton-03 Draft	15
10.11. Changes Since -02 Draft	15
10.12. Changes Since -01 Draft	15
10.13. Changes Since -00 Draft	15
11. References	16
11.1. Normative References	16
11.2. Informative References	18
Appendix A. Examples	19

A.1. REQUIRETLS SMTP Option	19
A.2. TLS-Required Header Field	20
Author's Address	21

1. Introduction

The SMTP [RFC5321] STARTTLS service extension [RFC3207] provides a means by which an SMTP server and client can establish a Transport Layer Security (TLS) protected session for the transmission of email messages. By default, TLS is used only upon mutual agreement (successful negotiation) of STARTTLS between the client and server; if this is not possible, the message is sent without transport encryption. Furthermore, it is common practice for the client to negotiate TLS even if the SMTP server's certificate is invalid.

Policy mechanisms such as DANE [RFC7672] and MTA-STS [RFC8461] may impose requirements for the use of TLS for email destined for some domains. However, such policies do not allow the sender to specify which messages are more sensitive and require transport-level encryption, and which ones are less sensitive and ought to be relayed even if TLS cannot be negotiated successfully.

The default opportunistic nature of SMTP TLS enables several "on the wire" attacks on SMTP security between MTAs. These include passive eavesdropping on connections for which TLS is not used, interference in the SMTP protocol to prevent TLS from being negotiated (presumably accompanied by eavesdropping), and insertion of a man-in-the-middle attacker exploiting the lack of server authentication by the client. Attacks are described in more detail in the Security Considerations section of this document.

REQUIRETLS consists of two mechanisms: an SMTP service extension and a message header field. The service extension is used to specify that a given message sent during a particular session **MUST** be sent over a TLS-protected session with specified security characteristics. It also requires that the SMTP server advertise that it supports REQUIRETLS, in effect promising that it will honor the requirement to enforce TLS transmission and REQUIRETLS support for onward transmission of those messages.

The TLS-Required message header field is used to convey a request to ignore recipient-side policy mechanisms such as MTA-STS and DANE, thereby prioritizing delivery over ability to negotiate TLS. Unlike the service extension, the TLS-Required header field allows the message to transit through one or more MTAs that do not support REQUIRETLS.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The formal syntax uses the Augmented Backus-Naur Form (ABNF) [RFC5234] including the core rules defined in Appendix B of that document.

2. The REQUIRETLS Service Extension

1. The textual name of the extension is "Require TLS".
2. The EHLO keyword value associated with this extension is "REQUIRETLS".
3. No additional SMTP verbs are defined by this extension.
4. One optional parameter ("REQUIRETLS") is added to the MAIL FROM command by this extension. No value is associated with this parameter.
5. The maximum length of a MAIL FROM command line is increased by 11 octets by the possible addition of a space and the REQUIRETLS keyword.
6. One new SMTP status code is defined by this extension to convey an error condition resulting from failure of the client to send to a server not also supporting the REQUIRETLS extension.
7. The REQUIRETLS extension is valid for message relay [RFC5321], submission [RFC6409], and the Local Mail Transfer Protocol (LMTP) [RFC2033]
8. The ABNF syntax for the MAIL FROM parameter is as follows:

```
requiretls-param = "REQUIRETLS"  
                  ; where requiretls-param is an instance of an  
                  ; esmtp-param used in Mail-parameters in  
                  ; RFC 5321 Section 4.1.2. There is no esmtp-value  
                  ; associated with requiretls-param.
```

In order to specify REQUIRETLS treatment for a given message, the REQUIRETLS option is specified on the MAIL FROM command when that message is transmitted. This option MUST only be specified in the

context of an SMTP session meeting the security requirements of REQUIRETLS:

- o The session itself MUST employ TLS transmission.
- o If the SMTP server to which the message is being transmitted is identified through an MX record lookup, its name MUST be validated via a DNSSEC signature on the recipient domain's MX record, or the MX hostname MUST be validated by an MTA-STS policy as described in Section 4.1 of RFC 8461 [RFC8461]. DNSSEC is defined in RFC 4033 [RFC4033], RFC 4034 [RFC4034], and RFC 4035 [RFC4035].
- o The certificate presented by the SMTP server MUST either verify successfully in a trust chain leading to a certificate trusted by the SMTP client or it MUST verify successfully using DANE as specified in RFC 7672 [RFC7672]. For trust chains, the choice of trusted (root) certificates is at the discretion of the SMTP client.
- o Following the negotiation of STARTTLS, the SMTP server MUST advertise in the subsequent EHLO response that it supports REQUIRETLS.

3. The TLS-Required Header Field

One new message header field [RFC5322], TLS-Required, is defined by this specification. It is used for messages for which the originator requests that recipient TLS policy (including MTA-STS [RFC8461] and DANE [RFC7672]) be ignored. This might be done, for example, to report a misconfigured mail server, such as an expired TLS certificate.

The TLS-Required header field has a single REQUIRED parameter:

- o No - The SMTP client SHOULD attempt to send the message regardless of its ability to negotiate STARTTLS with the SMTP server, ignoring policy-based mechanisms (including MTA-STS and DANE), if any, asserted by the recipient domain. Nevertheless, the client SHOULD negotiate STARTTLS with the server if available.

More than one instance of the TLS-Required header field MUST NOT appear in a given message.

The ABNF syntax for the TLS-Required header field is as follows:

```
requiretls-field = "TLS-Required:" [FWS] "No" CRLF
                  ; where requiretls-field in an instance of an
                  ; optional-field defined in RFC 5322 Section
                  ; 3.6.8.
FWS = <as defined in RFC 5322>
CRLF = <as defined in RFC 5322>
```

4. REQUIRETLS Semantics

4.1. REQUIRETLS Receipt Requirements

Upon receipt of the REQUIRETLS option on a MAIL FROM command during the receipt of a message, an SMTP server MUST tag that message as needing REQUIRETLS handling.

Upon receipt of a message not specifying the REQUIRETLS option on its MAIL FROM command but containing the TLS-Required header field in its message header, an SMTP server implementing this specification MUST tag that message with the option specified in the TLS-Required header field. If the REQUIRETLS MAIL FROM parameter is specified, the TLS-Required header field MUST be ignored but MAY be included in onward relay of the message.

The manner in which the above tagging takes place is implementation-dependent. If the message is being locally aliased and redistributed to multiple addresses, all instances of the message MUST be tagged in the same manner.

4.2. REQUIRETLS Sender Requirements

4.2.1. Sending with TLS Required

When sending a message tagged as requiring TLS for which the MAIL FROM return-path is not empty (an empty MAIL FROM return-path indicating a bounce message), the sending (client) MTA MUST:

1. Look up the SMTP server to which the message is to be sent as described in [RFC5321] Section 5.1.
2. If the server lookup is accomplished via the recipient domain's MX record (the usual case) and is not accompanied by a valid DNSSEC signature, the client MUST also validate the SMTP server name using MTA-STX as described in RFC 8461 [RFC8461] Section 4.1.
3. Open an SMTP session with the peer SMTP server using the EHLO verb.

4. Establish a TLS-protected SMTP session with its peer SMTP server and authenticate the server's certificate as specified in [RFC6125] or [RFC7672] as applicable. The hostname from the MX record lookup (or the domain name in the absence of an MX record where an A record is used directly) MUST match the DNS-ID or CN-ID of the certificate presented by the server.
5. Ensure that the response to the subsequent EHLO following establishment of the TLS protection advertises the REQUIRETLS capability.

The SMTP client SHOULD follow the recommendations in [RFC7525] or its successor with respect to negotiation of the TLS session.

If any of the above steps fail, the client MUST issue a QUIT to the server and repeat steps 2-5 with each host on the recipient domain's list of MX hosts in an attempt to find a mail path that meets the sender's requirements. The client MAY send other, unprotected, messages to that server if it has any prior to issuing the QUIT. If there are no more MX hosts, the client MUST NOT transmit the message to the domain.

Following such a failure, the SMTP client MUST send a non-delivery notification to the reverse-path of the failed message as described in section 3.6 of [RFC5321]. The following status codes [RFC5248] SHOULD be used:

- o REQUIRETLS not supported by server: 5.7.YYY REQUIRETLS needed
- o Unable to establish TLS-protected SMTP session: 5.7.10 Encryption needed

Refer to Section 5 for further requirements regarding non-delivery messages.

If all REQUIRETLS requirements have been met, transmit the message, issuing the REQUIRETLS option on the MAIL FROM command with the required option(s), if any.

4.2.2. Sending with TLS Optional

Messages tagged TLS-Required: No are handled as follows. When sending such a message, the sending (client) MTA MUST:

- o Look up the SMTP server to which the message is to be sent as described in [RFC5321] Section 5.1.

- o Open an SMTP session with the peer SMTP server using the EHLO verb. Attempt to negotiate STARTTLS if possible, and follow any policy published by the recipient domain, but do not fail if this is unsuccessful.

Some SMTP servers may be configured to require STARTTLS connections as a matter of policy and not accept messages in the absence of STARTTLS. A non-delivery notification **MUST** be returned to the sender if message relay fails due to an inability to negotiate STARTTLS when required by the server.

Since messages tagged with TLS-Required: No will sometimes be sent to SMTP servers not supporting REQUIRETLS, that option will not be uniformly observed by all SMTP relay hops.

4.3. REQUIRETLS Submission

An MUA or other agent making the initial introduction of a message has the option to decide whether to require TLS. If TLS is to be required, it **MUST** do so by negotiating STARTTLS and REQUIRETLS and include the REQUIRETLS option on the MAIL FROM command, as is done for message relay.

When TLS is not to be required, the sender **MUST** include the TLS-Required header field in the message. SMTP servers implementing this specification **MUST** interpret this header field as described in Section 4.1.

In either case, the decision whether to specify REQUIRETLS **MAY** be done based on a user interface selection or based on a ruleset or other policy. The manner in which the decision to require TLS is made is implementation-dependent and is beyond the scope of this specification.

4.4. Delivery of REQUIRETLS messages

Messages are usually retrieved by end users using protocols other than SMTP such as IMAP [RFC3501], POP [RFC1939], or web mail systems. Mail delivery agents supporting the REQUIRETLS SMTP option **SHOULD** observe the guidelines in [RFC8314].

5. Non-delivery message handling

Non-delivery ("bounce") messages usually contain important metadata about the message to which they refer, including the original message header. They therefore **MUST** be protected in the same manner as the original message. All non-delivery messages resulting from messages with the REQUIRETLS SMTP option, whether resulting from a REQUIRETLS

error or some other, MUST also specify the REQUIRETLS SMTP option unless redacted as described below.

The path from the origination of an error bounce message back to the MAIL FROM address may not share the same REQUIRETLS support as the forward path. Therefore, users requiring TLS are advised to make sure that they are capable of receiving mail using REQUIRETLS as well. Otherwise, such non-delivery messages will be lost.

If a REQUIRETLS message is bounced, the server MUST behave as if RET=HDRS was present as described in [RFC3461]. If both RET=FULL and REQUIRETLS are present, the RET=FULL MUST be disregarded. The SMTP client for a REQUIRETLS bounce message uses an empty MAIL FROM return-path as required by [RFC5321]. When the MAIL FROM return-path is empty, the REQUIRETLS parameter SHOULD NOT cause a bounce message to be discarded even if the next-hop relay does not advertise REQUIRETLS.

Senders of messages requiring TLS are advised to consider the possibility that bounce messages will be lost as a result of REQUIRETLS return path failure, and that some information could be leaked if a bounce message is not able to be transmitted with REQUIRETLS.

6. Reorigination considerations

In a number of situations, a mediator [RFC5598] originates a new message as a result of an incoming message. These situations include, but are not limited to, mailing lists (including administrative traffic such as message approval requests), Sieve [RFC5228], "vacation" responders, and other filters to which incoming messages may be piped. These newly originated messages may essentially be copies of the incoming message, such as with a forwarding service or a mailing list expander. In other cases, such as with a vacation message or a delivery notification, they will be different but might contain parts of the original message or other information for which the original message sender wants to influence the requirement to use TLS transmission.

Mediators that reoriginate messages should apply REQUIRETLS requirements in incoming messages (both requiring TLS transmission and requesting that TLS not be required) to the reoriginated messages to the extent feasible. A limitation to this might be that for a message requiring TLS, redistribution to multiple addresses while retaining the TLS requirement could result in the message not being delivered to some of the intended recipients.

User-side mediators (such as use of Sieve rules on a user agent) typically do not have access to the SMTP details, and therefore may not be aware of the REQUIRETLS requirement on a delivered message. Recipients that expect sensitive traffic should avoid the use of user-side mediators. Alternatively, if operationally feasible (such as when forwarding to a specific, known address), they should apply REQUIRETLS to all reoriginated messages that do not contain the "TLS-Required: No" header field.

7. IANA Considerations

If published as an RFC, this draft requests the addition of the following keyword to the SMTP Service Extensions Registry [MailParams]:

Textual name:	Require TLS
EHLO keyword value:	REQUIRETLS
Syntax and parameters:	(no parameters)
Additional SMTP verbs:	none
MAIL and RCPT parameters:	REQUIRETLS parameter on MAIL
Behavior:	Use of the REQUIRETLS parameter on the MAIL verb causes that message to require the use of TLS and tagging with REQUIRETLS for all onward relay.
Command length increment:	11 characters

If published as an RFC, this draft requests the addition of an entry to the Simple Mail Transfer Protocol (SMTP) Enhanced Status Codes Registry [SMTPStatusCodes]:

Code:	5.7.YYY
Sample Text:	REQUIRETLS support required
Associated basic status code:	550
Description:	This indicates that the message was not able to be forwarded because it was received with a REQUIRETLS requirement and none of the SMTP servers to which the message should be forwarded provide this support.
Reference:	(this document)
Submitter:	J. Fenton
Change controller:	IESG

If published as an RFC, this draft requests the addition of an entry to the Permanent Message Header Field Names Registry [PermMessageHeaderFields]:

Header field name: TLS-Required
Applicable protocol: mail
Status: standard
Author/change controller: IETF
Specification document: (this document)

This section is to be updated for publication by the RFC Editor.

8. Security Considerations

The purpose of REQUIRETLS is to give the originator of a message control over the security of email they send, either by conveying an expectation that it will be transmitted in an encrypted form "over the wire" or explicitly that transport encryption is not required if it cannot be successfully negotiated.

The following considerations apply to the REQUIRETLS service extension but not the TLS-Required header field, since messages specifying the header field are less concerned with transport security.

8.1. Passive attacks

REQUIRETLS is generally effective against passive attackers who are merely trying to eavesdrop on an SMTP exchange between an SMTP client and server. This assumes, of course, the cryptographic integrity of the TLS connection being used.

8.2. Active attacks

Active attacks against TLS encrypted SMTP connections can take many forms. One such attack is to interfere in the negotiation by changing the STARTTLS command to something illegal such as XXXXXXXX. This causes TLS negotiation to fail and messages to be sent in the clear, where they can be intercepted. REQUIRETLS detects the failure of STARTTLS and declines to send the message rather than send it insecurely.

A second form of attack is a man-in-the-middle attack where the attacker terminates the TLS connection rather than the intended SMTP server. This is possible when, as is commonly the case, the SMTP client either does not verify the server's certificate or establishes the connection even when the verification fails. REQUIRETLS requires successful certificate validation before sending the message.

Another active attack involves the spoofing of DNS MX records of the recipient domain. An attacker having this capability could potentially cause the message to be redirected to a mail server under

the attacker's own control, which would presumably have a valid certificate. REQUIRETLS requires that the recipient domain's MX record lookup be validated either using DNSSEC or via a published MTA-STS policy that specifies the acceptable SMTP server hostname(s) for the recipient domain.

8.3. Bad Actor MTAs

A bad-actor MTA along the message transmission path could misrepresent its support of REQUIRETLS and/or actively strip REQUIRETLS tags from messages it handles. However, since intermediate MTAs are already trusted with the cleartext of messages they handle, and are not part of the threat model for transport-layer security, they are also not part of the threat model for REQUIRETLS.

It should be reemphasized that since SMTP TLS is a transport-layer security protocol, messages sent using REQUIRETLS are not encrypted end-to-end and are visible to MTAs that are part of the message delivery path. Messages containing sensitive information that MTAs should not have access to MUST be sent using end-to-end content encryption such as OpenPGP [RFC4880] or S/MIME [RFC8551].

8.4. Policy Conflicts

In some cases, the use of the TLS-Required header field may conflict with a recipient domain policy expressed through the DANE [RFC7672] or MTA-STS [RFC8461] protocols. Although these protocols encourage the use of TLS transport by advertising availability of TLS, the use of "TLS-Required: No" header field represents an explicit decision on the part of the sender not to require the use of TLS, such as to overcome a configuration error. The recipient domain has the ultimate ability to require TLS by not accepting messages when STARTTLS has not been negotiated; otherwise, "TLS-Required: No" is effectively directing the client MTA to behave as if it does not support DANE nor MTA-STS.

9. Acknowledgements

The author would like to acknowledge many helpful suggestions on the ietf-smtp and uta mailing lists, in particular those of Viktor Dukhovni, Chris Newman, Tony Finch, Jeremy Harris, Arvel Hathcock, John Klensin, Barry Leiba, John Levine, Rolf Sonneveld, and Per Thorsheim.

10. Revision History

To be removed by RFC Editor upon publication as an RFC.

10.1. Changes since -08 Draft

Additional changes in response to IESG review:

- o Unify wording describing TLS-Required in Appendix A.2.
- o Add specifics on verification of mail server hostnames with certificates.
- o Wording tweak in 4.3 to emphasize optional nature of REQUIRETLS.
- o Update S/MIME reference from RFC 5751 to 8551

10.2. Changes since -07 Draft

Changes in response to IESG review and IETF Last Call comments:

- o Change associated status code for 5.7.YYY from 530 to 550.
- o Correct textual name of extension in IANA Considerations for consistency with the rest of the document.
- o Remove special handling of bounce messages in Section 4.1.
- o Change name of header field from RequireTLS to TLS-Required and make capitalization of parameter consistent.
- o Remove mention of transforming RET=FULL to RET=HDRS on relay in Section 5.
- o Replace Section 6 dealing with mailing lists with a more general section on reorigination by mediators.
- o Add security considerations section on policy conflicts.

10.3. Changes since -06 Draft

Various changes in response to AD review:

- o Reference RFC 7525 for TLS negotiation recommendations.
- o Make reference to requested 5.7.YYY error code consistent.
- o Clarify applicability to LMTP and submission.

- o Provide ABNF for syntax of SMTP option and header field and examples in Appendix A.
- o Correct use of normative language in Section 5.
- o Clarify case where REQUIRETLS option is used on bounce messages.
- o Improve Security Requirements wording to be inclusive of both SMTP option and header field.

10.4. Changes since -05 Draft

Corrected IANA Permanent Message Header Fields Registry request.

10.5. Changes since -04 Draft

Require validation of SMTP server hostname via DNSSEC or MTA-STS policy when TLS is required.

10.6. Changes since -03 Draft

Working Group Last Call changes, including:

- o Correct reference for SMTP DANE
- o Clarify that RequireTLS: NO applies to both MTA-STS and DANE policies
- o Correct newly-defined status codes
- o Update MTA-STS references to RFC

10.7. Changes since -02 Draft

- o More complete documentation for IANA registration requests.
- o Changed bounce handling to use RET parameters of [RFC3461], along with slightly more liberal transmission of bounces even if REQUIRETLS can't be negotiated.

10.8. Changes since -01 Draft

- o Converted DEEP references to RFC 8314.
- o Removed REQUIRETLS options: CHAIN, DANE, and DNSSEC.
- o Editorial corrections, notably making the header field name consistent (RequireTLS rather than Require-TLS).

10.9. Changes since -00 Draft

- o Created new header field, Require-TLS, for use by "NO" option.
- o Removed "NO" option from SMTP service extension.
- o Recommend DEEP requirements for delivery of messages requiring TLS.
- o Assorted copy edits

10.10. Changes since fenton-03 Draft

- o Wording improvements from Rolf Sonneveld review 22 July 2017
- o A few copy edits
- o Conversion from individual to UTA WG draft

10.11. Changes Since -02 Draft

- o Incorporation of "MAY TLS" functionality as REQUIRETLS=NO per suggestion on UTA WG mailing list.
- o Additional guidance on bounce messages

10.12. Changes Since -01 Draft

- o Specified retries when multiple MX hosts exist for a given domain.
- o Clarified generation of non-delivery messages
- o Specified requirements for application of REQUIRETLS to mail forwarders and mailing lists.
- o Clarified DNSSEC requirements to include MX lookup only.
- o Corrected terminology regarding message retrieval vs. delivery.
- o Changed category to standards track.

10.13. Changes Since -00 Draft

- o Conversion of REQUIRETLS from an SMTP verb to a MAIL FROM parameter to better associate REQUIRETLS requirements with transmission of individual messages.

- o Addition of an option to require DNSSEC lookup of the remote mail server, since this affects the common name of the certificate that is presented.
- o Clarified the wording to more clearly state that TLS sessions must be established and not simply that STARTTLS is negotiated.
- o Introduced need for minimum encryption standards (key lengths and algorithms)
- o Substantially rewritten Security Considerations section

11. References

11.1. Normative References

[MailParams]

Internet Assigned Numbers Authority (IANA), "IANA Mail Parameters", 2007,
<<http://www.iana.org/assignments/mail-parameters>>.

[PermMessageHeaderFields]

Internet Assigned Numbers Authority (IANA), "Permanent Message Header Field Names Registry", 2004,
<<https://www.iana.org/assignments/message-headers/message-headers.xhtml#perm-headers>>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.

[RFC3207] Hoffman, P., "SMTP Service Extension for Secure SMTP over Transport Layer Security", RFC 3207, DOI 10.17487/RFC3207, February 2002, <<https://www.rfc-editor.org/info/rfc3207>>.

[RFC3461] Moore, K., "Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)", RFC 3461, DOI 10.17487/RFC3461, January 2003,
<<https://www.rfc-editor.org/info/rfc3461>>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005,
<<https://www.rfc-editor.org/info/rfc4033>>.

- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <<https://www.rfc-editor.org/info/rfc4034>>.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<https://www.rfc-editor.org/info/rfc4035>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.
- [RFC5248] Hansen, T. and J. Klensin, "A Registry for SMTP Enhanced Mail System Status Codes", BCP 138, RFC 5248, DOI 10.17487/RFC5248, June 2008, <<https://www.rfc-editor.org/info/rfc5248>>.
- [RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <<https://www.rfc-editor.org/info/rfc5321>>.
- [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <<https://www.rfc-editor.org/info/rfc5322>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC7672] Dukhovni, V. and W. Hardaker, "SMTP Security via Opportunistic DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS)", RFC 7672, DOI 10.17487/RFC7672, October 2015, <<https://www.rfc-editor.org/info/rfc7672>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8314] Moore, K. and C. Newman, "Cleartext Considered Obsolete: Use of Transport Layer Security (TLS) for Email Submission and Access", RFC 8314, DOI 10.17487/RFC8314, January 2018, <<https://www.rfc-editor.org/info/rfc8314>>.
- [RFC8461] Margolis, D., Risher, M., Ramakrishnan, B., Brotman, A., and J. Jones, "SMTP MTA Strict Transport Security (MTA-STS)", RFC 8461, DOI 10.17487/RFC8461, September 2018, <<https://www.rfc-editor.org/info/rfc8461>>.
- [SMTPStatusCodes]
Internet Assigned Numbers Authority (IANA), "Simple Mail Transfer Protocol (SMTP) Enhanced Status Codes Registry", 2008, <<http://www.iana.org/assignments/smtp-enhanced-status-codes>>.

11.2. Informative References

- [RFC1939] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, DOI 10.17487/RFC1939, May 1996, <<https://www.rfc-editor.org/info/rfc1939>>.
- [RFC2033] Myers, J., "Local Mail Transfer Protocol", RFC 2033, DOI 10.17487/RFC2033, October 1996, <<https://www.rfc-editor.org/info/rfc2033>>.
- [RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, DOI 10.17487/RFC3501, March 2003, <<https://www.rfc-editor.org/info/rfc3501>>.
- [RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <<https://www.rfc-editor.org/info/rfc4880>>.
- [RFC5228] Guenther, P., Ed. and T. Showalter, Ed., "Sieve: An Email Filtering Language", RFC 5228, DOI 10.17487/RFC5228, January 2008, <<https://www.rfc-editor.org/info/rfc5228>>.
- [RFC5598] Crocker, D., "Internet Mail Architecture", RFC 5598, DOI 10.17487/RFC5598, July 2009, <<https://www.rfc-editor.org/info/rfc5598>>.

- [RFC6409] Gellens, R. and J. Klensin, "Message Submission for Mail", STD 72, RFC 6409, DOI 10.17487/RFC6409, November 2011, <<https://www.rfc-editor.org/info/rfc6409>>.
- [RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <<https://www.rfc-editor.org/info/rfc8551>>.

Appendix A. Examples

This section is informative.

A.1. REQUIRETLS SMTP Option

The TLS-Required SMTP option is used to express the intent of the sender that the associated message be relayed using TLS. In the following example, lines beginning with C: are transmitted from the SMTP client to the server, and lines beginning with S: are transmitted in the opposite direction.

```
S: 220 mail.example.net ESMTP
C: EHLO mail.example.org
S: 250-mail.example.net Hello example.org [192.0.2.1]
S: 250-SIZE 52428800
S: 250-8BITMIME
S: 250-PIPELINING
S: 250-STARTTLS
S: 250 HELP
C: STARTTLS
S: TLS go ahead
```

(at this point TLS negotiation takes place. The remainder of this session occurs within TLS.)

```
S: 220 mail.example.net ESMTP
C: EHLO mail.example.org
S: 250-mail.example.net Hello example.org [192.0.2.1]
S: 250-SIZE 52428800
S: 250-8BITMIME
S: 250-PIPELINING
S: 250-REQUIRETLS
S: 250 HELP
C: MAIL FROM:<roger@example.org> REQUIRETLS
S: 250 OK
C: RCPT TO:<editor@example.net>
S: 250 Accepted
C: DATA
S: 354 Enter message, ending with "." on a line by itself
```

(message follows)

```
C: .
S: 250 OK
C: QUIT
```

A.2. TLS-Required Header Field

The TLS-Required header field is used when the sender requests that the mail system not heed a default policy of the recipient domain requiring TLS. It might be used, for example, to allow problems with the recipient domain's TLS certificate to be reported:

From: Roger Reporter <roger@example.org>
To: Andy Admin <admin@example.com>
Subject: Certificate problem?
TLS-Required: No
Date: Fri, 18 Jan 2019 10:26:55 -0800
Message-ID: <5c421a6f79c0e_d153ff8286d45c468473@mail.example.org>

Andy, there seems to be a problem with the TLS certificate
on your mail server. Are you aware of this?

Roger

Author's Address

Jim Fenton
Altmode Networks
Los Altos, California 94024
USA

Email: fenton@bluepopcorn.net