Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: August 17, 2017

J. Fenton
Altmode Networks

February 13, 2017

SMTP Require TLS Option
draft-fenton-smtp-require-tls-03

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 prioritized over security. This document describes a complementary SMTP service extension, REQUIRETLS. If the REQUIRETLS option 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 policy mechanisms such as SMTP STS and DANE be ignored when relaying a high priority message.

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 August 17, 2017.

Copyright Notice

Table of Contents

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 fails to authenticate it.

Policy mechanisms such as DANE [RFC7672] and SMTP STS [I-D.ietf-uta-mta-sts] 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 urgent 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 followed by eavesdropping), and insertion of a man-in-the-middle attacker taking advantage of the lack of server authentication by the client. Attacks are more described in more detail in the Security Considerations section of this document.

The REQUIRETLS SMTP service extension allows the SMTP client to specify that a given message sent during a particular session MUST be sent over a TLS protected session with specified security characteristics, or conversely that delivery should be prioritized over ability to negotiate TLS. For messages requiring TLS negotiation, it also requires that the SMTP server advertise that it also supports REQUIRETLS, in effect promising that it will honor the requirement to require TLS transmission and REQUIRETLS support for onward transmission of messages specifying that requirement.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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. One MAIL FROM option is defined by this extension.

4. Two new SMTP status codes are defined by this extension to convey error conditions resulting from failure of the client to negotiate a TLS connection with the required security and as a result of an attempt to send to a server not also supporting the REQUIRETLS extension.

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. With the exception of REQUIRETLS=NO (described below), this option MUST only be specified in the context of an SMTP session meeting the security requirements that have been specified:

- The session itself MUST employ TLS transmission, unless the NO parameter is specified.

- Any server authentication requirements specified as an option to the REQUIRETLS option (see below) MUST have been satisfied in establishing the current session.

An optional parameter to the REQUIRETLS MAIL FROM option specifies the requirements for server authentication that MUST be used for any onward transmission of the following message. The parameter takes the form of either a single value or comma-separated list, separated from the REQUIRETLS option by a single "=" (equals-sign) character. If present, the parameter MUST take one or more of the following values:

- CHAIN - The certificate presented by the SMTP server MUST verify successfully in a trust chain leading to a certificate trusted by the SMTP client. The choice of trusted (root) certificates by the client is at their own discretion. The client MAY choose to use the certificate set maintained by the CA/B forum [citation needed] for this purpose.

- DANE - The certificate presented by the SMTP server MUST verify succesfully using DANE as specified in RFC 7672 [RFC7672].

- DNSSEC - The server MUST confirm that any MX record or CNAME lookup used to locate the SMTP server must be DNSSEC [RFC4035] signed and valid.

- 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, if any, asserted by the recipient domain. Nevertheless, the client MAY negotiate STARTTLS with the server if available. If the NO parameter is present, any other REQUIRETLS parameter MUST NOT be used.

The CHAIN and DANE parameters are additive; if both are specified, either method of certificate validation is acceptable. If neither CHAIN nor DANE is specified, the certificate presented by the SMTP server is not required to be verified.

3. REQUIRETLS Semantics

3.1. 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 with the specified option(s). The manner in which this 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.

3.2. REQUIRETLS Sender Requirements

3.2.1. Sending with TLS Required

When sending a message tagged with REQUIRETLS other than the REQUIRETLS=NO option, the sending (client) MTA MUST:

1. Look up the SMTP server to which the message is to be sent. If the DNSSEC option is included in the message tag, the MX record lookups in this process MUST use DNSSEC verification and the response(s) MUST be DNSSEC-signed in order to ensure the integrity of the resource identifier [RFC6125] used to authenticate the SMTP server.

2. Open an SMTP session with the peer SMTP server using the EHLO verb. The server MUST advertise the REQUIRETLS capability.

3. Establish a TLS-protected SMTP session with its peer SMTP server and authenticate the server’s certificate with the specified authentication method.

4. The SMTP client SHOULD also require that meaningfully secure cipher algorithms and key lengths be negotiated with the server. The choices of key lengths and algorithms change over time, so a specific requirement is not presented here.

If any of the above steps fail, the client SHOULD issue a QUIT to the server and repeat steps 2-4 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. If there are no more MX hosts or if the MX record lookup is not DNSSEC-protected and DNSSEC verification is required, the client MUST NOT transmit the message and MUST issue an SMTP QUIT command to the server. The client MAY send other, unprotected, messages to that server prior to issuing the QUIT if it has any.

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:

- DNSSEC lookup failure: 5.x.x DNSSEC lookup required
- REQUIRETLS not supported by server: 5.7.x REQUIRETLS needed
- Unable to establish TLS-protected SMTP session: 5.7.10 Encryption needed

Refer to Section 4. 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.

3.2.2. Sending with TLS Optional

Messages tagged REQUIRETLS=NO are handled differently from other REQUIRETLS messages, as follows. When sending a message tagged with REQUIRETLS=NO, the sending (client) MTA MUST:

- Look up the SMTP server to which the message is to be sent as described in [RFC5321].
- 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.
- If the server does not advertise the REQUIRETLS capability, send the message in the usual manner (without the REQUIRETLS option, because the server will not understand the option).
- If the server advertises the REQUIRETLS capability, send the message with the REQUIRETLS=NO option.

Some SMTP servers that are configured to expect STARTTLS connections as a matter of policy may not accept messages in the absence of STARTTLS. This MUST be expected, and a non-delivery notification returned to the sender.

Messages tagged with the REQUIRETLS=NO option will be sent without the option to SMTP servers not supporting REQUIRETLS. REQUIRETLS=NO MAY therefore not persist through multiple email relay hops.

3.3. REQUIRETLS Submission

An MUA or other agent making the initial introduction of a message to SMTP has authority to decide whether to require TLS, and if so, using what authentication method(s). It does so by issuing the REQUIRETLS option in the MAIL FROM command during message submission. This MAY be done based on a user interface selection, on a header field included in the message, or based on policy. The manner in which the decision to require TLS is made is implementation-dependent and is beyond the scope of this specification.

3.4. 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 REQUIRETLS SHOULD require that retrieval of messages requiring transport encryption take place over authenticated, encrypted channels.

4. 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, whether resulting from a REQUIRETLS error or some other, MUST employ REQUIRETLS using the same authentication method(s) as the message that caused the error to occur.

It should be noted that 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 of REQUIRETLS (other than REQUIRETLS=NO) are advised to make sure that they are capable of receiving mail using REQUIRETLS at the same authentication method(s) as messages they send. Otherwise, such non-delivery messages will be lost.

If unable to send a bounce message due to a REQUIRETLS failure (the return path not supporting the TLS requirements in the original message), the MTA sending the bounce message MAY send a redacted non-delivery message to the postmaster of the domain identified in the envelope-From address identifying the message only by Message-ID and indicating the type of failure. The original From, Return-path, To, Sender, Cc, and related header fields MUST NOT be included in this message.

Senders of messages specifying REQUIRETLS (other than REQUIRETLS=NO) are advised to consider the increased likelihood that bounce messages will be lost as a result of REQUIRETLS return path failure.

5. Mailing list considerations

Mailing lists, upon receipt of a message, originate new messages to list addresses, as distinct from an aliasing operation that redirects the original message, in some cases to multiple recipients. The requirement to preserve the REQUIRETLS tag and options therefore does not extend to mailing lists. REQUIRETLS users SHOULD use caution when sending to mailing lists and MUST NOT assume that REQUIRETLS applies to messages from the list operator to list members.

Mailing list operators MAY apply REQUIRETLS requirements in incoming messages to the resulting messages they originate. If this is done, they SHOULD also apply these requirements to administrative traffic, such as messages to moderators requesting approval of messages.

6. IANA Considerations

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

If published as an RFC, this draft also requests the creation of a registry, REQUIRETLS Security Requirements, to be initially populated with the CHAIN, DANE, DNSSEC, and NO keywords.

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] in the 5.7.YYY range to indicate lack of REQUIRETLS support by an SMTP server to which a message is being routed.

This section is to be removed during conversion into an RFC by the RFC Editor.

7. Security Considerations

The purpose of REQUIRETLS is to improve communications security for email by giving the originator of a message an expectation that it will be transmitted in an encrypted form "over the wire". When used, REQUIRETLS changes the traditional behavior of email transmission, which favors delivery over the ability to send email messages using transport-layer security, to one in which requested security takes precedence over delivery and domain-level policy.

The following considerations apply to STARTTLS other than the STARTTLS=NO option, since messages specifying that option are specifying less concern with transport security.

7.1. 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.

7.2. 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. The REQUIRETLS CHAIN and DANE options allow the message sender to specify that successful certificate validation, using either or both of two different methods, is required before sending the message.

Another active attack involves the spoofing of DNS MX records of the recipient domain. An attacker having this capability could cause the message to be redirected to a mail server under the attacker's own control, which would presumably have a valid certificate. The REQUIRETLS DNSSEC option allows the message sender to require that valid DNSSEC [RFC4033] signatures be obtained when locating the recipient's mail server, in order to address that attack.

In addition to support of the DNSSEC option, domains receiving email SHOULD deploy DNSSEC and SMTP clients SHOULD deploy DNSSEC verification.

7.3. 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 [RFC5751].

8. Acknowledgements

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

9. Revision History

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

9.1. Changes Since -02 Draft

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

9.2. Changes Since -01 Draft

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

9.3. Changes Since -00 Draft

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

- 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.

- Clarified the wording to more clearly state that TLS sessions must be established and not simply that STARTTLS is negotiated.

- Introduced need for minimum encryption standards (key lengths and algorithms)

- Substantially rewritten Security Considerations section

10. References

10.1. Normative References

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://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, <http://www.rfc-editor.org/info/rfc3207>.

[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, <http://www.rfc-editor.org/info/rfc4035>.

[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, <http://www.rfc-editor.org/info/rfc5248>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <http://www.rfc-editor.org/info/rfc5321>.

[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>.

10.2. Informative References

[I-D.ietf-uta-mta-sts]
Margolis, D., Risher, M., Ramakrishnan, B., Brotman, A., and J. Jones, "SMTP MTA Strict Transport Security (MTA-STS)", draft-ietf-uta-mta-sts-02 (work in progress), December 2016.

[RFC1939] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, DOI 10.17487/RFC1939, May 1996, <http://www.rfc-editor.org/info/rfc1939>.

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, DOI 10.17487/RFC3501, March 2003, <http://www.rfc-editor.org/info/rfc3501>.

[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, <http://www.rfc-editor.org/info/rfc4033>.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <http://www.rfc-editor.org/info/rfc4880>.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, DOI 10.17487/RFC5751, January 2010, <http://www.rfc-editor.org/info/rfc5751>.

[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, <http://www.rfc-editor.org/info/rfc6125>.

[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, <http://www.rfc-editor.org/info/rfc7672>.

Author's Address

Jim Fenton
Altmode Networks
704 Benvenue Avenue
Los Altos, California 94024
USA

Email: fenton@bluepopcorn.net

Network Working Group
Internet-Draft
Updates: 1939, 2595, 3464, 3501, 5068,
6186, 6409 (if approved)
Intended status: Standards Track
Expires: September 14, 2017

K. Moore
Network Heretics
C. Newman
Oracle

March 13, 2017

Mail User Agent Strict Transport Security (MUA-STS)
draft-ietf-uta-email-deep-06

Abstract

This specification defines a set of requirements and facilities designed to improve email confidentiality between a mail user agent (MUA) and a mail submission or mail access server. This provides mechanisms intended to increase use of already deployed Transport Layer Security (TLS) technology and provides a model for a mail user agent's confidentiality assurance. This enables mail service providers to advertise strict transport security (STS) policies that request MUAs increase confidentiality assurance.

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 September 14, 2017.

Table of Contents

1. Introduction

Software that provides email service via Internet Message Access Protocol (IMAP) [RFC3501], Post Office Protocol (POP) [RFC1939] and/or Simple Mail Transfer Protocol (SMTP) Submission [RFC6409] usually has Transport Layer Security (TLS) [RFC5246] support but often does not use it in a way that maximizes end-user confidentiality. This specification proposes changes to email software and deployments intended to increase the use of TLS and record when that use occurs. This adapts the strict transport security (STS) model described in [RFC6797] to cover mail user agents (MUAs).

In brief, this memo now recommends that:

- MUAs associate a minimum confidentiality assurance level with each mail account, and disconnections associated with that account that do not provide the minimum confidentiality assurance level associated with that account.

- By default, MUAs assign a minimum confidentiality assurance level that requires use of TLS with certificate validation for all TCP connections;

- o TLS on a well-known port ("Implicit TLS") be supported for IMAP, POP, and SMTP Submission [RFC6409] for all electronic mail user agents (MUAs), servers, and service providers;

- o MUAs and mail protocol servers cooperate (via mechanisms defined in this specification) to upgrade security feature use and record/indicate that usage appropriately. The security upgrade model is aligned with the HTTP STS specification [RFC6797].

This does not address use of TLS with SMTP for message relay (where Message Submission [RFC6409] does not apply). Improved use of TLS with SMTP for message relay requires a different approach. One approach to address that topic is described in [RFC7672].

The recommendations in this memo do not replace the functionality of, and are not intended as a substitute for, end-to-end encryption of electronic mail.

This draft is subject to change. Implementation of this proposal is not recommended at this time. Please discuss this proposal on the ietf-uta mailing list.

2. Conventions and Terminology Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This specification expresses syntax using the Augmented Backus-Naur Form (ABNF) as described in [RFC5234], including the core rules in Appendix B and rules from [RFC5322].

In examples, "C:" and "S:" indicate lines sent by the client and server respectively. If a single "C:" or "S:" label applies to multiple lines, then the line breaks between those lines are for editorial clarity only and are not part of the actual protocol exchange.

3. Mail Account Confidentiality Assurance Level

A "mail account" refers to the network services an end user uses to read, submit and manage email communications on the Internet. This typically involves at least one mail access server (IMAP or POP) and at least one SMTP submission server. An end user uses a mail user agent (MUA) to access a mail account. (Most MUAs support the ability to access multiple mail accounts.) This document uses the term "confidentiality assurance level" to indicate the degree to which the network connections between an MUA and a mail account have

confidentiality protection from both passive and active attackers on the network.

The configuration necessary for a mail account includes an email address, connection information, and authentication credentials for network services. MUAs compliant with this specification MUST also associate a minimum confidentiality assurance level with each mail account. If during a session with a network service, the requirements for the minimum confidentiality assurance level associated with that mail account are not met, the MUA MUST NOT continue the session with the network service. MUAs MUST support at least the ability to detect whether a session with a network service implements confidentiality assurance level 1 as described in the next section. Note that the minimum confidentiality assurance level associated with an account applies to all protocol interactions and all servers associated with the account.

MUAs SHOULD continuously indicate to the user the current confidentiality assurance level of any account currently in use when reading, submitting and managing mail (e.g., via a lock icon, background colors, or other indications similar to those commonly used in web browsers for a similar purpose) and SHOULD indicate the minimum confidentiality assurance level for each account whenever displaying a list of mail accounts. Note that the displayed confidentiality assurance level for a current session could be higher than the minimum confidentiality assurance level set at account configuration, but never lower. If multiple active connections are associated with an account or view, the indication of the current confidentiality assurance level associated with the account should reflect the level provided by the least confidential connection. It is therefore possible that at any given instant some services associated with a mail account meet the minimum confidentiality assurance level associated with the account, and other services do not. An MUA MAY continue to interact with those services for which the minimum confidentiality assurance level is met, while refusing to interact with those services for which the minimum confidentiality assurance level is not met. For example, if the IMAP service associated with a mail account meets the minimum confidentiality assurance level, but the Mail Submission service associated with that account does not, the MUA MAY continue to permit reading mail from that account but MUST NOT send mail until it can do so using a Submission service that meets the minimum confidentiality assurance level for that account.

Account configuration occurs when an MUA is first used to access a particular service, when a user wishes to access or submit mail through servers in addition to those specified or found during first use, or when a user explicitly requests to change account

configuration parameters such as server names, user names, passwords, client certificates, etc. Account configuration can be entirely manual (entering server names explicitly) or partially automated via a mechanism such as DNS SRV records [RFC6186]. MUAs SHOULD require a minimum confidentiality assurance level of 1 as the default for newly configured accounts.

This document defines two initial confidentiality assurance levels, 1 and 0. It is expected that other levels may be defined in the future, as needed to thwart increasingly sophisticated and/or pervasive attacks.

3.1. Confidentiality Assurance Level 1

A mail account has a confidentiality assurance level of 1 when the following conditions are met on all TCP server connections associated with an account. This includes connections to POP, IMAP and SMTP submission servers as well as any other associated protocols defined now or in the future. Examples of protocols associated with a mail account include managesieve [RFC5804] and MTQP [RFC3887].

- TCP connections MUST successfully negotiate TLS via either Implicit TLS Section 4 or STARTTLS.
- For protocols using TCP, both client and server must support, and negotiate, a TLS version of 1.1 or greater.
- MUAs MUST implement [RFC7817] and PKIX [RFC5280].
- MUAs MAY implement DANE [RFC6698] as an alternate means of verifying TLS server certificates. For confidentiality assurance level 1, a certificate may be considered valid if it can be validated using either DANE or PKIX.
- User agents MUST abort a TLS session if the TLS negotiation fails or the server’s certificate or identity fails to verify. A user may reconfigure the account to lower the expected level of confidentiality if he/she chooses. Reduction of expected account confidentiality MUST NOT be done on a click-through basis.

The end user is part of the system that protects the user’s confidentiality and security. As a result, it’s critical not to present the end user with a simple action that reduces their confidentiality in response to certificate validation failure. An MUA which offers a user actions such as "connect anyway", "trust certificate for future connections" or "lower confidentiality assurance for this account" in response to certificate validation failure is not implementing a minimum confidentiality assurance of 1

as defined in this section and thus does not comply with this document. Examples of acceptable actions to offer would be "work offline", "try again later", and "open service provider status web page".

3.2. Confidentiality Assurance Level 0

MUAs MAY support the ability to configure accounts with a minimum confidentiality assurance level of 0. At this level, the MUA MUST attempt to negotiate TLS, but MAY ignore server certificate validation failures. MUAs MAY support use of connections without TLS, or using TLS versions prior to TLS 1.1, for accounts with a minimum confidentiality assurance level of 0. Even for accounts with a minimum confidentiality assurance level of 0, MUAs SHOULD attempt TLS first if available, and MUST implement the ability to reconnect without TLS if TLS negotiation fails for reasons other than server certificate validity.

Note that if TLS is not used, or a version of TLS prior to TLS 1.1 is negotiated, or the TLS server certificate is not successfully validated as described in Section 3.1, the client MUST clearly indicate to the user that there is currently no assurance of confidentiality for the mail account or connection.

3.3. Other Confidentiality Assurance Levels

This specification is not intended to limit experimentation and innovation with respect to user confidentiality. As a result, an implementation MAY implement confidentiality assurance levels other than those defined in this document, as long as those levels are distinguished in user interfaces from those defined in this document, and the ordering associated with them reflects the actual expectation of confidentiality provided. However, implementation of levels below confidentiality assurance level 0, as described in the previous section, is discouraged. Implementers are also cautioned that end users may be confused by too many confidentiality assurance levels.

As stated above, higher confidentiality assurance levels may be standardized in the future. For example, a future confidentiality assurance levels might require multiple independent trust anchors for server certificate validation.

4. Implicit TLS

Previous standards for use of email protocols with TLS used the STARTTLS mechanism: [RFC2595], [RFC3207], and [RFC3501]. With STARTTLS, the client establishes a clear text application session and determines whether to issue a STARTTLS command based on server

capabilities and client configuration. If the client issues a STARTTLS command, a TLS handshake follows that can upgrade the connection. While this mechanism has been deployed, an alternate mechanism where TLS is negotiated immediately at connection start on a separate port (referred to in this document as "Implicit TLS") has been deployed more successfully. To increase use of TLS, this specification recommends use of implicit TLS by new POP, IMAP and SMTP Submission software.

4.1. Implicit TLS for POP

When a TCP connection is established for the "pop3s" service (default port 995), a TLS handshake begins immediately. Clients MUST implement the certificate validation mechanism described in [RFC7817]. Once the TLS session is established, POP3 [RFC1939] protocol messages are exchanged as TLS application data for the remainder of the TCP connection. After the server sends a +OK greeting, the server and client MUST enter AUTHORIZATION state, even if client credentials were supplied during the TLS handshake.

See Section 11.1.1 for additional information on client certificate authentication. See Section 13.3 for port registration information.

4.2. Implicit TLS for IMAP

When a TCP connection is established for the "imaps" service (default port 993), a TLS handshake begins immediately. Clients MUST implement the certificate validation mechanism described in [RFC3501] and SHOULD implement the certificate validation mechanism described in [RFC7817]. Once the TLS session is established, IMAP [RFC3501] protocol messages are exchanged as TLS application data for the remainder of the TCP connection. If client credentials were provided during the TLS handshake that the server finds acceptable, the server MAY issue a PREAUTH greeting in which case both the server and client enter AUTHENTICATED state. If the server issues an OK greeting then both server and client enter NOT AUTHENTICATED state.

See Section 11.1.1 for additional information on client certificate authentication. See Section 13.4 for port registration information.

4.3. Implicit TLS for SMTP Submission

When a TCP connection is established for the "submissions" service (default port 465), a TLS handshake begins immediately. Clients MUST implement the certificate validation mechanism described in [RFC7817]. Once a TLS session is established, message submission protocol data [RFC6409] is exchanged as TLS application data for the remainder of the TCP connection. (Note: the "submissions" service

name is defined in section 10.3 of this document, and follows the usual convention that the name of a service layered on top of Implicit TLS consists of the name of the service as used without TLS, with an "s" appended.)

The STARTTLS mechanism on port 587 is relatively widely deployed due to the situation with port 465 (discussed in Section 13.5). This differs from IMAP and POP services where implicit TLS is more widely deployed on servers than STARTTLS. It is desirable to migrate core protocols used by MUA software to implicit TLS over time for consistency as well as the additional reasons discussed in Appendix A. However, to maximize use of encryption for submission it is desirable to support both mechanisms for Message Submission over TLS for a transition period of several years. As a result, clients and servers SHOULD implement both STARTTLS on port 587 and implicit TLS on port 465 for this transition period. Note that there is no significant difference between the security properties of STARTTLS on port 587 and implicit TLS on port 465 if the implementations are correct and both client and server are configured to require successful negotiation of TLS prior to message submission (as required in Section 11.1).

Note that the submissions port provides access to a Mail Submission Agent (MSA) as defined in [RFC6409] so requirements and recommendations for MSAs in that document apply to the submissions port, including the requirement to implement SMTP AUTH [RFC4954].

See Section 11.1.1 for additional information on client certificate authentication. See Section 13.5 for port registration information.

4.4. Implicit TLS Connection Closure for POP, IMAP and SMTP

When a client or server wishes to close the connection, it SHOULD initiate the exchange of TLS close alerts before TCP connection termination. The client MAY, after sending a TLS close alert, gracefully close the TCP connection without waiting for a TLS response from the server.

5. Email Security Upgrading Using Security Directives

Once an improved email security mechanism is deployed and ready for general use, it is desirable to continue using it for all future email service. For example, TLS is widely deployed in email software, but use of TLS is often not required. At the time this is written, deployed mail user agents (MUAs) [RFC5598] usually make a determination if TLS is available when an account is first configured and may require use of TLS with that account if and only if it was initially available. If the service provider makes TLS available

after initial client configuration, many MUAs will not notice the change.

Alternatively, a security feature may be purely opportunistic and thus subject to downgrade attacks. For example, at the time this was written, most TLS stacks that support TLS 1.2 will use an older TLS version if the peer does not support TLS 1.2 and many do so without alerting the user of the reduced security. Thus a variety of active attacks could cause the loss of TLS 1.2 benefits. Only if client policy is upgraded to require TLS 1.2 can the client prevent all downgrade attacks. However, this sort of security policy upgrade will be ignored by most users unless it is automated.

This section describes a mechanism, called "security directives", which is designed to permit an MUA to recognize when a service provider has committed to provide certain server security features, and that it’s safe for the client to change its configuration for that account to require that such features be present in future sessions with that server. Once the client has changed the configuration for a mail service to require specific server security features, those features are said to be "latched".

Note that security directives are a separate mechanism from minimum confidentiality assurance levels. A connection between a client and a service MUST meet the requirements of both the minimum confidentiality assurance level associated with the account, and the conditions of any security directives established for that service. Otherwise the client MUST abandon the connection. When an MUA implements both minimum confidentiality assurance levels and security directives, then both the end-user and the service provider independently have the ability to improve the end-user’s confidentiality.

A security directive has the following formal syntax:

```
directive       =  directive-name [ "=" directive-value ]

directive-name  =  token

directive-value =  token

token           =  <As defined in RFC 7230>
```

This is a subset of the syntax used by HSTS [RFC6797] as revised in [RFC7230]; but simplified for use by protocols other than HTTP.

6. Server Strict Transport Security Policy

Servers supporting this extension MUST advertise an STS policy. This includes a list of security directives the server administrator has explicitly configured as recommended for use by clients (the list MAY be empty). When a server advertises a security directive associated with a security facility, it is making a commitment to support that facility (or a revised version of that facility) indefinitely and recommending that the client save that directive with the account configuration and require that security facility for future connections to that server.

Server STS policy may also include a "sts-url" directive with a value containing an https Uniform Resource Locator (URL) [RFC2818] that the client can save and subsequently resolve for the user in the event of a security connection problem. Server STS policy has the following formal syntax:

```
sts-policy        =  [directive *(";" [SP] directive)]
```

Protocol extensions to advertise STS policy for email servers are defined in Section 9.

The IANA Considerations Section 13 defines a registry so that more directives can be defined in the future. Three initial directives are defined for use by MUAs in Section 13.2: tls-version, sts-url, and tls-cert.

7. Client Storage of Email Security Directives

Before a client can consider storing any security directives, it MUST verify that the connection to the server uses TLS, the server has been authenticated, and any requirements for any previously saved security directives are met. Then the client performs the following steps for each security directive in the STS policy:

1. If the security directive name is not known to the client, skip to the next directive.

2. If the security directive is already saved with the same value (or a value considered greater than the current value in the directive’s definition), the client skips the security directive and moves on to the next one.

3. The client verifies the connection meets the requirements of the security directive. If the connection does not, then the directive will not be saved. For example, a security directive claiming that the server supports tls-version 1.2 will not be

saved by a client if the currently negotiated TLS session is using TLS 1.1.

4. If previous steps pass, the client SHOULD update the current account configuration to save the security directive.

Once a security directive is saved, all subsequent connections to that host require any associated security feature. For this confidentiality protection to work as desired clients MUST NOT offer a click-through-to-connect action when unable to achieve connection security matching the saved security directives.

7.1. Security Directive Upgrade Example

Suppose a server advertises the "tls-version" directive name with value "1.1". A client that successfully negotiates either TLS 1.1 or TLS 1.2 SHOULD save this directive. The server may subsequently change the value to "1.2". When a client with "1.1" saved value connects and negotiates TLS 1.2, it will upgrade the saved directive value to "1.2". However, a client that only supports TLS 1.1 will continue to require use of TLS 1.1 and work with that server as long as it permits TLS 1.1. This way individual clients can require the newer/stronger protocol (e.g., TLS 1.2), while older clients can continue to communicate securely (albeit potentially less so) using the older protocol.

7.2. Security Policy Failures

When a security directive has been saved for connections from a client to a server and the facility identified by that directive is no longer available, this results in a connection failure. An MUA SHOULD inform the user of a potential threat to their confidentiality and offer to resolve a previously-recorded sts-url https URL if one is available. MUAs are discouraged from offering a lightweight option to reset or ignore directives as this defeats the benefit they provide to end users.

8. Recording TLS Cipher Suite in Received Header

The ESMTPS transmission type [RFC3848] provides trace information that can indicate TLS was used when transferring mail. However, TLS usage by itself is not a guarantee of confidentiality or security. The TLS cipher suite provides additional information about the level of security made available for a connection. This defines a new SMTP "tls" Received header additional-registered-clause that is used to record the TLS cipher suite that was negotiated for the connection. The value included in this additional clause SHOULD be the registered cipher suite name (e.g., TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256)

included in the TLS cipher suite registry. In the event the implementation does not know the name of the cipher suite (a situation that should be remedied promptly), a four-digit hexadecimal cipher suite identifier MAY be used. The ABNF for the field follows:

```
tls-cipher-clause  =  CFWS "tls" FWS tls-cipher

tls-cipher         =  tls-cipher-suite-name / tls-cipher-suite-hex

tls-cipher-name    =  ALPHA *(ALPHA / DIGIT / "_")
                      ; as registered in IANA cipher suite registry

tls-cipher-hex     =  "0x" 4HEXDIG
```

9. Extensions for STS Policy and Reporting

This memo defines optional mechanisms for use by MUAs to communicate saved STS policy to servers and for servers to advertise policy. One purpose of such mechanisms is to permit servers to determine which and how many clients have saved security directives, and thus, to permit operators to be aware of potential impact to their users should support for such facilities be changed. For IMAP, the existing ID command is extended to provide this capability. For SMTP Submission, a new CLIENT command is defined. No similar mechanism is defined for POP in this version of the memo to keep POP simpler, but one may be added in the future if deemed necessary.

In addition, for each of IMAP, POP, and SMTP, a new STS capability is defined so the client can access the server’s STS policy.

9.1. IMAP STS Extension

When an IMAP server advertises the STS capability, that indicates the IMAP server implements IMAP4 ID [RFC2971] with additional field values defined here. This is grouped with the ID command because that is the existing IMAP mechanism for clients to report data for server logging, and provides a way for the server to report the STS policy.

sts From server to client, the argument to this ID field is the server STS policy. Servers MUST provide this information in response to an ID command.

saved From client to server, this is a list of security directives the client has saved for this server (the client MAY omit the value for the sts-url directive in this context). Servers MAY record this information so administrators know the expected security properties of the client and can thus act to avoid

security policy failures (e.g., by renewing server certificates on time, etc).

policy-fail From client to server, a list including one or more security directives the client has saved that the client was unable to achieve. This allows clients to report errors to the server prior to terminating the connection in the event an acceptable security level is unavailable.

directives From client to server, this is a list of security directive names the client supports that are not saved.

tls Server-side IMAP proxies that accept TLS connections from clients and connect in-the-clear over a fully private secure network to the server SHOULD use this field to report the tls-cipher (syntax as defined in Section 8) to the server.

IMAP clients SHOULD use the IMAP ID command to report policy failures and determine the server STS policy. Clients MAY use the ID command to report other security directive information. IMAP servers MUST implement the ID command at least to report STS policy to clients.

```
<client connected to port 993 and negotiated TLS successfully>
S: * OK [CAPABILITY IMAP4rev1 STS ID AUTH=PLAIN
        AUTH=SCRAM-SHA-1] hello
C: a001 ID ("name" "Demo Mail" "version" "1.5" "saved"
        "tls-version=1.1; tls-cert"
        "directives" "tls-version=1.2")
S: * ID ("name" "Demo Server" "version" "1.7" "sts-policy"
         "tls-version=1.1; tls-cert;
         sts-url=https://www.example.com/security-support.html")
S: a001 OK ID completed
```

Example 1

This example shows a client that successfully negotiated TLS version 1.1 or later and verified the server’s certificate as required by IMAP. Even if the client successfully validates the server certificate, it will not require tls-version 1.2 in the future as the server does not advertise that version as policy. The client has not yet saved an STS URL, but if the client successfully validated the server certificate, it will save the provided URL.

```
<client connected to port 993 and negotiated TLS successfully>
S: * OK [CAPABILITY IMAP4rev1 DEEP ID AUTH=PLAIN
        AUTH=SCRAM-SHA-1] hello
C: a001 ID ("name" "Demo Mail" "version" "1.5" "policy-failure"
        "tls-cert=pkix")
S: * ID ("name" "Demo Server" "version" "1.7" "sts-policy"
         "tls-version=1.1;
          sts-url=<https://www.example.com/security-support.html>")
S: a001 OK ID completed
C: a002 LOGOUT
```

Example 2

This example shows a client that negotiated TLS, but was unable to verify the server's certificate using PKIX. The policy-failure informs the server of this problem, at which point the client can disconnect. If the client had previously saved the sts-url security directive from this server, it could offer to resolve that URI. However, the sts-policy in this exchange is ignored due to the failure to meet the conditions of the tls-version security directive.

```
<IMAP Proxy connected over private network on port 143, there is
a client connected to the proxy on port 993 that negotiated TLS>
S: * OK [CAPABILITY IMAP4rev1 DEEP ID AUTH=PLAIN
        AUTH=SCRAM-SHA-1] hello
C: a001 ID ("name" "Demo Mail" "version" "1.5" "saved"
        "tls-version=1.1; tls-cert=pkix"
        "tls" "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256")
S: * ID ("name" "Demo Server" "version" "1.7" "sts-policy"
         "tls-version=1.1; tls-cert=pkix;
         sts-url=https://www.example.com/support.html")
S: a001 OK ID completed
```

Example 3

This example shows the connection from an IMAP proxy to a back-end server. The client connected to the proxy and sent the ID command shown in example 1, and the proxy has added the "tls" item to the ID command so the back-end server can log the cipher suite that was used on the connection from the client.

9.2. POP DEEP Extension

POP servers supporting this specification MUST implement the POP3 extension mechanism [RFC2449]. POP servers MUST advertise the DEEP capability with an argument indicating the server's DEEP status. (Note: DEEP is an ancronym for the original name of this

specification, before the terms were changed to align better with those used in HSTS.)

```
<client connected to port 995 and negotiated TLS successfully>
S: +OK POP server ready
C: CAPA
S: +OK Capability list follows
S: TOP
S: SASL PLAIN SCRAM-SHA-1
S: RESP-CODES
S: PIPELINING
S: UIDL
S: STS tls-version=1.2
   sts-url=<https://www.example.com/security-support.html>
S: .
```

Example 4

After verifying the TLS server certificate and issuing CAPA, the client can save any or all of the STS policy. If the client connects to this same server later and has a security failure, the client can direct the user's browser to the previously-saved URL where the service provider can provide advice to the end user.

9.3. SMTP MSTS Extension

SMTP Submission servers supporting this specification MUST implement the MSTS SMTP extension. The name of this extension is MSTS. The EHLO keyword value is MSTS and the sts-policy ABNF is the syntax of the EHLO keyword parameters. This does not add parameters to the MAIL FROM or RCPT TO commands. This also adds a CLIENT command to SMTP which is used to report client information to the server. The formal syntax for the command follows:

```
deep-cmd            = "CLIENT" 1*(SP deep-parameter)

deep-parameter      = name / version / policy-fail
                      / directives / tls / future-extension

name                = "name" SP esmtp-value

version             = "version" SP esmtp-value

saved               = "saved" SP directive-list

policy-fail         = "policy-fail" SP directive-list

directive-list      = DQUOTE [directive
                      *(";" [SP] directive)] DQUOTE

directives          = "directives" SP directive-list

tls                 = "tls" SP tls-cipher

future-extension    = Atom SP String

Atom                = <as defined in RFC 5321>

String              = <as defined in RFC 5321>
```

The CLIENT command parameters listed here have the same meaning as the parameters used in the IMAP STS extension (Section 9.1). The server responds to the CLIENT command with a "250" if the command has correct syntax and a "501" if the command has incorrect syntax.

```
<client connected to port 465 and negotiated TLS successfully>
S: 220 example.com Demo SMTP Submission Server
C: EHLO client.example.com
S: 250-example.com
S: 250-8BITMIME
S: 250-PIPELINING
S: 250-DSN
S: 250-AUTH PLAIN LOGIN
S: 250-MSTS tls-version=1.2; tls-cert;
   sts-url=<https://www.example.com/status.html>
S: 250-BURL imap
S: 250 SIZE 0
C: CLIENT name demo_submit version 1.5 saved "tls-version=1.1;
   tls-cert=pkix+dane" directives "tls-version=1.2"
S: 250 OK
```

Example 5

10. Account Setup Considerations

10.1. Use of SRV records in Establishing Configuration

This section updates [RFC6186] by changing the preference rules and adding a new SRV service label _submissions._tcp to refer to Message Submission with implicit TLS.

User-configurable MUAs SHOULD support use of [RFC6186] for account setup. However, when using configuration information obtained by this method, MUAs SHOULD default to a minimum confidentiality assurance level of 1, unless the user has explicitly requested reduced confidentiality. This will have the effect of causing the MUA to ignore advertised configurations that do not support TLS, even when those advertised configurations have a higher priority than other advertised configurations.

When using [RFC6186] configuration information, Mail User Agents SHOULD NOT automatically establish new configurations that do not require TLS for all servers, unless there are no advertised configurations using TLS. If such a configuration is chosen, prior to attempting to authenticate to the server or use the server for message submission, the MUA SHOULD warn the user that traffic to that server will not be encrypted and that it will therefore likely be intercepted by unauthorized parties. The specific wording is to be determined by the implementation, but it should adequately capture the sense of risk given the widespread incidence of mass surveillance of email traffic.

When establishing a new configuration for connecting to an IMAP, POP, or SMTP Submission server, an MUA SHOULD NOT blindly trust SRV records unless they are signed by DNSSEC and have a valid signature. Instead, the MUA SHOULD warn the user that the DNS-advertised mechanism for connecting to the server is not authenticated, and request the user to manually verify the connection details by reference to his or her mail service provider’s documentation.

Similarly, an MUA MUST NOT consult SRV records to determine which servers to use on every connection attempt, unless those SRV records are signed by DNSSEC and have a valid signature. However, an MUA MAY consult SRV records from time to time to determine if an MSP’s server configuration has changed, and alert the user if it appears that this has happened. This can also serve as a means to encourage users to upgrade their configurations to require TLS if and when their MSPs support it.

10.2. Certificate Pinning

During account setup, the MUA will identify servers that provide account services such as mail access and mail submission (the previous section describes one way to do this). The certificates for these servers are verified using the rules described in [RFC7817] and PKIX [RFC5280]. In the event the certificate does not validate due to an expired certificate, lack of appropriate chain of trust or lack of identifier match, the MUA MAY create a persistent binding between that certificate and the saved host name for the server. This is called certificate pinning. Certificate pinning is only appropriate during account setup and MUST NOT be offered in response to a failed certificate validation for an existing account. An MUA that allows certificate pinning MUST NOT allow a certificate pinned for one account to validate connections for other accounts.

A pinned certificate is subject to a man-in-the-middle attack at account setup time, and lacks a mechanism to revoke or securely refresh the certificate. Therefore use of a pinned certificate does not meet the requirement for a minimum confidentiality assurance level of 1, and an MUA MUST NOT indicate a confidentiality assurance level of 1 for an account or connection using a pinned certificate. Additional advice on certificate pinning is present in [RFC6125].

11. Implementation Requirements

This section details requirements for implementations of electronic mail protocol clients and servers. A requirement for a client or server implementation to support a particular feature is not the same thing as a requirement that a client or server running a conforming implementation be configured to use that feature. Requirements for Mail Service Providers (MSPs) are distinct from requirements for protocol implementations, and are listed in a separate section.

11.1. All Implementations (Client and Server)

These requirements apply to MUAs as well as POP, IMAP and SMTP Submission servers.

- All implementations MUST implement TLS 1.2 or later, and be configurable to support implicit TLS using the TLS 1.2 protocol or later [RFC5246].

- All implementations MUST implement the recommended cipher suites described in [RFC7525] or a future BCP or standards track revision of that document.

- All implementations MUST be configurable to require TLS before performing any operation other than capability discovery and STARTTLS.

- The IMAP specification [RFC3501] is hereby modified to revoke the second paragraph of section 11.1 and replace it with the text from the first three bullet items in this list. See Appendix B of [RFC7817] to see additional modifications to IMAP certificate validation rules.

- The standard for use of TLS with IMAP, POP3 and ACAP [RFC2595] is modified to revoke section 2.1 and replace it with the text from the first three bullet items in this list. See Appendix B of [RFC7817] to see additional modifications to RFC 2595 certificate validation rules.

- The standard for Message Submission [RFC6409] is updated to add the first three bullet items above to section 4.3 as well as to require implementation of the TLS server identity check as described in [RFC7817] and PKIX [RFC5280].

11.1.1. Client Certificate Authentication

MUAs and mail servers MAY implement client certificate authentication on the implicit TLS port. Servers MUST NOT request a client certificate during the TLS handshake unless the server is configured to accept some client certificates as sufficient for authentication and the server has the ability to determine a mail server authorization identity matching such certificates. How to make this determination is presently implementation specific. Clients MUST NOT provide a client certificate during the TLS handshake unless the server requests one and the client has determined the certificate can be safely used with that specific server, OR the client has been explicitly configured by the user to use that particular certificate with that server. How to make this determination is presently implementation specific. If the server accepts the client's certificate as sufficient for authorization, it MUST enable the SASL EXTERNAL [RFC4422] mechanism. An IMAPS server MAY issue a PREAUTH greeting instead of enabling SASL EXTERNAL. A client supporting client certificate authentication with implicit TLS MUST implement the SASL EXTERNAL [RFC4422] mechanism using the appropriate authentication command (AUTH for POP3 [RFC5034], AUTH for SMTP Submission [RFC4954], AUTHENTICATE for IMAP [RFC3501]).

11.2. Mail Server Implementation Requirements

These requirements apply to servers that implement POP, IMAP or SMTP Submission.

- Servers MUST implement the appropriate STS Policy and Reporting extensions described in Section 9

- IMAP and SMTP submission servers SHOULD implement and be configurable to support STARTTLS. This enables discovery of new TLS availability, and can increase usage of TLS by legacy clients.

- Servers MUST NOT advertise STARTTLS capability if it is unlikely to succeed based on server configuration (e.g., there is no server certificate installed).

- SMTP message submission servers that have negotiated TLS SHOULD add a Received header field to the message including the tls clause described in Section 8.

- Servers MUST be configurable to include the TLS cipher information in any connection or user logging or auditing facility they provide.

11.3. Mail User Agent Implementation Requirements

This section describes requirements on Mail User Agents (MUAs) using IMAP, POP, and/or Submission protocols. Note: Requirements pertaining to use of Submission servers are also applicable when using SMTP servers (e.g., port 25) for mail submission.

- User agents SHOULD indicate to users at configuration time, the minimum expected level of confidentiality based on appropriate security inputs such as which security directives are pre-set, the number of trust anchors, certificate validity, use of an extended validation certificate, TLS version supported, and TLS cipher suites supported by both server and client. This indication SHOULD also be present when editing or viewing account configuration.

- For any mail service not initially configured to require TLS, MUAs SHOULD detect when STARTTLS and/or implicit TLS becomes available for a protocol and set the tls-version security directive if the server advertises the tls-version=1.1 or higher security policy after a successful negotiation (including certificate validation) of TLS 1.1.

- Whenever requested to establish any configuration that does not require both TLS and server certificate verification to talk to a server or account, an MUA SHOULD warn its user that his or her mail traffic (including password, if applicable) will be exposed to attackers, and give the user an opportunity to abort the connection prior to transmission of any such password or traffic.

- MUAs SHOULD support the ability to save the "tls-version=1.2" security directive (the TLS library has to provide an API that controls permissible TLS versions, and communicates the negotiated TLS protocol version to the application, for this to be possible).

- See Section 3 for additional requirements.

11.4. Non-configurable MUAs and nonstandard access protocols

MUAs which are not configurable to use user-specified servers MUST implement TLS or similarly other strong encryption mechanism when communicating with their mail servers. This generally applies to MUAs that are pre-configured to operate with one or more specific services, whether or not supplied by the vendor of those services.

MUAs using protocols other than IMAP, POP, and Submission to communicate with mail servers, MUST implement TLS or other similarly robust encryption mechanism in conjunction with those protocols.

11.5. Compliance for Anti-Virus/Anti-Spam Software and Services

There are multiple ways to connect an Anti-Virus and/or Anti-Spam (AVAS) service to a mail server. Some mechanisms, such as the de-facto milter protocol, are out of scope for this specification. However, some services use an SMTP relay proxy that intercepts mail at the application layer to perform a scan and proxy or forward to another MTA. Deploying AVAS services in this way can cause many problems [RFC2979] including direct interference with this specification, and other forms of confidentiality or security reduction. An AVAS product or service is considered compliant with this specification if all IMAP, POP and SMTP-related software (including proxies) it includes are compliant with this specification, and each of these services advertise and support all security directives that the actual end-servers advertise.

Note that end-to-end email encryption prevents AVAS software and services from using email content as part of a spam or virus assessment. Furthermore, while a minimum confidentiality assurance level of 1 or better can prevent a man-in-the-middle from introducing spam or virus content between the MUA and Submission server, it does

not prevent other forms of client or account compromise. Use of AVAS services for submitted email therefore remains necessary.

12. Mail Service Provider Requirements

This section details requirements for providers of IMAP, POP, and/or SMTP submission services, for providers who claim to conform to this specification.

12.1. Server Requirements

Mail Service Providers MUST use server implementations that conform to this specification.

12.2. MSPs MUST provide Submission Servers

This document updates the advice in [RFC5068] by making Implicit TLS on port 465 the preferred submission port.

Mail Service Providers that accept mail submissions from end-users using the Internet Protocol MUST provide one or more SMTP Submission services, separate from the SMTP MTA services used to process incoming mail. Those submission services MUST be configured to support Implicit TLS on port 465 and SHOULD support STARTTLS if port 587 is used.

MSPs MAY also support submission of messages via one or more designated SMTP servers to facilitate compatibility with legacy MUAs.

Discussion: SMTP servers used to accept incoming mail or to relay mail are expected to accept mail in cleartext. This is incompatible with the purpose of this memo which is to encourage encryption of traffic between mail servers. There is no such requirement for mail submission servers to accept mail in cleartext or without authentication. For other reasons, use of separate SMTP submission servers has been best practice for many years.

12.3. TLS Server Certificate Requirements

MSPs MUST maintain valid server certificates for all servers. See [RFC7817] for the recommendations and requirements necessary to achieve this.

If a protocol server provides service for more than one mail domain, it MAY use a separate IP address for each domain and/or a server certificate that advertises multiple domains. This will generally be necessary unless and until it is acceptable to impose the constraint that the server and all clients support the Server Name Indication

extension to TLS [RFC6066]. For more discussion of this problem, see section 5.1 of [RFC7817].

12.4. Recommended DNS records for mail protocol servers

This section discusses not only the DNS records that are recommended, but also implications of DNS records for server configuration and TLS server certificates.

12.4.1. MX records

It is recommended that MSPs advertise MX records for handling of inbound mail (instead of relying entirely on A or AAAA records), and that those MX records be signed using DNSSEC. This is mentioned here only for completeness, as handling of inbound mail is out of scope for this document.

12.4.2. SRV records

MSPs SHOULD advertise SRV records to aid MUAs in determination of proper configuration of servers, per the instructions in [RFC6186].

MSPs SHOULD advertise servers that support Implicit TLS in preference to those which support cleartext and/or STARTTLS operation.

12.4.3. DNSSEC

All DNS records advertised by an MSP as a means of aiding clients in communicating with the MSP’s servers, SHOULD be signed using DNSSEC.

12.4.4. TLSA records

MSPs SHOULD advertise TLSA records to provide an additional trust anchor for public keys used in TLS server certificates. However, TLSA records MUST NOT be advertised unless they are signed using DNSSEC.

12.5. MSP Server Monitoring

MSPs SHOULD regularly and frequently monitor their various servers to make sure that: TLS server certificates remain valid and are not about to expire, TLSA records match the public keys advertised in server certificates, are signed using DNSSEC, server configurations are consistent with SRV advertisements, and DNSSEC signatures are valid and verifiable. Failure to detect expired certificates and DNS configuration errors in a timely fashion can result in significant loss of service for an MSP’s users and a significant support burden for the MSP.

12.6. Advertisement of STS policies

MSPs SHOULD advertise STS policies that include at least tls11, tls-cert and sts-url, with the latter having an associated https URL that can be used to inform clients of service outages or problems impacting client confidentiality. Note that advertising tls-cert is a commitment to maintain and renew server certificates. A MSP MAY also specifically indicate a commitment to support PKIX validation, DANE validation, or both, using tls-cert=pkix, tls-cert=dane, or tls-cert=pkix+dane, respectively.

12.7. Require TLS

New servers and services SHOULD be configured to require TLS unless it’s necessary to support legacy clients or existing client configurations.

12.8. Changes to Internet Facing Servers

When an MSP changes the Internet Facing Servers providing mail access and mail submission services, including SMTP-based spam/virus filters, it is generally necessary to support the same and/or a newer version of TLS and the same security directives that were previously advertised.

13. IANA Considerations

13.1. Security Directive Registry

IANA shall create (has created) the registry "STS Security Directives". This registry is a single table and will use an expert review process [RFC5226]. Each registration will contain the following fields:

Name: The name of the security directive. This follows the directive-name ABNF.

Value: The permitted values of the security directive. This should also explain if the value is optional or mandatory and what to do if the value is not recognized.

Description: This describes the meaning of the security directive and the conditions under which the directive is saved.

Scope: The protocols to which this security directive applies. Presently this may be MSTS (for MUA STS), HSTS (for HTTP STS), or ALL.

Intended Usage: One of COMMON, LIMITED USE or OBSOLETE.

Reference: Optional reference to specification.

Submitter: The identify of the submitter or submitters.

Change Controller: The identity of the change controller for the registration. This will be "IESG" in case of registrations in IETF-produced documents.

The expert reviewer will verify the directive name follows the ABNF, and that the value and description fields are clear, unambiguous, do not overlap existing deployed technology, do not create security problems and appropriately considers interoperability issues. Email security directives intended for LIMITED USE have a lower review bar (interoperability and overlap issues are less of a concern). The reviewer may approve a registration, reject for a stated reason or recommend the proposal have standards track review due to importance or difficult subtleties.

Standards-track registrations may be updated if the relevant standards are updated as a consequence of that action. Non-standards-track entries may be updated by the listed change controller. The entry’s name and submitter may not be changed. In exceptional cases, any aspect of any registered entity may be updated at the direction of the IESG (for example, to correct a conflict).

13.2. Initial Set of Security Directives

This document defines three initial security directives for the registry as follows, and registers the two additional directives specified in [RFC6797].

Name: tls-version

Value: Mandatory; 1.1 refers to [RFC4346] or later and 1.2 refers to [RFC5246] or later. Future versions may be added; this is ignored if the version is unrecognized.

Description: This directive indicates that the TLS version negotiated must be the specified version or later. In the event this directive is saved and only an older TLS version is available, that results in STS policy failure.

Scope: MUA only

Intended Usage: COMMON

Reference: RFC XXXX (this document once published)

Submitter: Authors of this document

Change Controller: IESG

Name: tls-cert

Value: Optional; pkix refers to PKIX certificate validation; dane refers to DANE certificate validation; pkix+dane refers to use of both PKIX and DANE validation; any refers to any validation method the client considers acceptable. If no value is supplied, "any" is assumed.

Description: This directive indicates that TLS was successfully negotiated and the server certificate was successfully verified by the client [RFC5280] and the server certificate identity was verified using the algorithm appropriate for the protocol (see Section 4). This directive is saved if the client sees this in the advertised server STS policy after successfully negotiating TLS and verifying the certificate and server identity using a means consistent with the associated (or implied) value. Note that an advertisement of either tls-cert=pkix or tls-cert=pkix+dane in a server's STS policy indicates that the server commits to using certificates that are verifiable using PKIX in the future, but tls-cert=pkix implies no commitment regarding DANE support. Similarly, an advertisement of either tls-cert=dane or tls-cert=pkix+dane indicates that the server commits to using certificates that are verifiable using DANE in the future, but tls-cert=dane implies no commitment regarding PKIX support. An advertisement of tls-cert or tls-cert=any indicates only that the server will continue to provide valid server certificates, but makes no commitment about the means of verifiability. (For the HSTS protocol, the presence of a Strict-Transport-Security response header serves as an indication that the certificate should be valid, so the tls-cert directive is never specified in that protocol.)

Scope: MUA only

Intended Usage: COMMON

Reference: RFC XXXX (this document once published)

Submitter: Authors of this document

Change Controller: IESG

Name: sts-url

Value: Mandatory for server-policy, optional for client reporting. The value is an https URL.

Description: This directive indicates that the client SHOULD resolve (with appropriate certificate validation) and display the URL in the event of a policy failure.

Scope: MUA only

Intended Usage: COMMON

Reference: RFC XXXX (this document once published)

Submitter: Authors of this document

Change Controller: IESG

Name: max-age

Value: see [RFC6797].

Description: see [RFC6797].

Scope: HSTS only

Intended Usage: COMMON

Reference: [RFC6797]

Submitter: Authors of this document

Change Controller: IESG

Name: includeSubDomains

Value: None

Description: see [RFC6797].

Scope: HSTS only

Intended Usage: COMMON

Reference: [RFC6797]

Submitter: Authors of this document

Change Controller: IESG

13.3. POP3S Port Registration Update

IANA is asked to update the registration of the TCP well-known port 995 using the following template ([RFC6335]):

Service Name: pop3s
Transport Protocol: TCP
Assignee: IETF <iesg@ietf.org>
Contact: IESG <iesg@ietf.org>
Description: POP3 over TLS protocol
Reference: RFC XXXX (this document once published)
Port Number: 995

13.4. IMAPS Port Registration Update

IANA is asked to update the registration of the TCP well-known port 993 using the following template ([RFC6335]):

Service Name: imaps
Transport Protocol: TCP
Assignee: IETF <iesg@ietf.org>
Contact: IESG <iesg@ietf.org>
Description: IMAP over TLS protocol
Reference: RFC XXXX (this document once published)
Port Number: 993

13.5. Submissions Port Registration

IANA is asked to assign an alternate usage of port 465 in addition to the current assignment using the following template ([RFC6335]):

Service Name: submissions
Transport Protocol: TCP
Assignee: IETF <iesg@ietf.org>
Contact: IESG <iesg@ietf.org>
Description: Message Submission over TLS protocol
Reference: RFC XXXX (this document once published)
Port Number: 465

This is a one time procedural exception to the rules in RFC 6335. This requires explicit IESG approval and does not set a precedent. Historically, port 465 was briefly registered as the "smtps" port. This registration made no sense as the SMTP transport MX infrastructure has no way to specify a port so port 25 is always used. As a result, the registration was revoked and was subsequently reassigned to a different service. In hindsight, the "smtps"

registration should have been renamed or reserved rather than revoked. Unfortunately, some widely deployed mail software interpreted "smtps" as "submissions" [RFC6409] and used that port for email submission by default when an end-user requests security during account setup. If a new port is assigned for the submissions service, email software will either continue with unregistered use of port 465 (leaving the port registry inaccurate relative to de-facto practice and wasting a well-known port), or confusion between the de-facto and registered ports will cause harmful interoperability problems that will deter use of TLS for message submission. The authors believe both of these outcomes are less desirable than a wart in the registry documenting real-world usage of a port for two purposes. Although STARTTLS-on-port-587 has deployed, it has not replaced deployed use of implicit TLS submission on port 465.

13.6. STS IMAP Capability

This document adds the STS capability to the IMAP capabilities registry. This is described in Section 9.1.

13.7. STS POP3 Capability

This document adds the STS capability to the POP3 capabilities registry.

CAPA Tag: STS

Arguments: sts-policy

Added Commands: none

Standard Commands affected: none

Announced status / possible differences: both / may change after STLS

Commands Valid in States: N/A

Specification Reference: This document

Discussion: See Section 9.2.

13.8. MSTS SMTP EHLO Keyword

This document adds the MSTS EHLO Keyword to the SMTP Service Extension registry. This is described in Section 9.3.

13.9. MAIL Parameters Additional-registered-clauses Sub-Registry

This document adds the following entry to the "Additional-registered-clauses" sub-registry of the "MAIL Parameters" registry, created by [RFC5321]:

Clause Name: tls

Description: Indicates the TLS cipher suite used for a transport connection.

Syntax Summary: See tls-cipher ABNF Section 8

Reference: This document.

14. Security Considerations

This entire document is about security considerations. In general, this is targeted to improve mail confidentiality and to mitigate threats external to the email system such as network-level snooping or interception; this is not intended to mitigate active attackers who have compromised service provider systems.

It could be argued that sharing the name and version of the client software with the server has privacy implications. Although providing this information is not required, it is encouraged so that mail service providers can more effectively inform end-users running old clients that they need to upgrade to protect their security, or know which clients to use in a test deployment prior to upgrading a server to have higher security requirements.

15. References

15.1. Normative References

[RFC1939] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, DOI 10.17487/RFC1939, May 1996, <http://www.rfc-editor.org/info/rfc1939>.

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

[RFC2449] Gellens, R., Newman, C., and L. Lundblade, "POP3 Extension Mechanism", RFC 2449, DOI 10.17487/RFC2449, November 1998, <http://www.rfc-editor.org/info/rfc2449>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[RFC2971] Showalter, T., "IMAP4 ID extension", RFC 2971, DOI 10.17487/RFC2971, October 2000, <http://www.rfc-editor.org/info/rfc2971>.

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

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, DOI 10.17487/RFC3501, March 2003, <http://www.rfc-editor.org/info/rfc3501>.

[RFC5034] Siemborski, R. and A. Menon-Sen, "The Post Office Protocol (POP3) Simple Authentication and Security Layer (SASL) Authentication Mechanism", RFC 5034, DOI 10.17487/RFC5034, July 2007, <http://www.rfc-editor.org/info/rfc5034>.

[RFC5068] Hutzler, C., Crocker, D., Resnick, P., Allman, E., and T. Finch, "Email Submission Operations: Access and Accountability Requirements", BCP 134, RFC 5068, DOI 10.17487/RFC5068, November 2007, <http://www.rfc-editor.org/info/rfc5068>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://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, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[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, <http://www.rfc-editor.org/info/rfc5280>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <http://www.rfc-editor.org/info/rfc5321>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <http://www.rfc-editor.org/info/rfc5322>.

[RFC6186] Daboo, C., "Use of SRV Records for Locating Email Submission/Access Services", RFC 6186, DOI 10.17487/RFC6186, March 2011, <http://www.rfc-editor.org/info/rfc6186>.

[RFC6409] Gellens, R. and J. Klensin, "Message Submission for Mail", STD 72, RFC 6409, DOI 10.17487/RFC6409, November 2011, <http://www.rfc-editor.org/info/rfc6409>.

[RFC6797] Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", RFC 6797, DOI 10.17487/RFC6797, November 2012, <http://www.rfc-editor.org/info/rfc6797>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[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, <http://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, <http://www.rfc-editor.org/info/rfc7672>.

[RFC7817] Melnikov, A., "Updated Transport Layer Security (TLS) Server Identity Check Procedure for Email-Related Protocols", RFC 7817, DOI 10.17487/RFC7817, March 2016, <http://www.rfc-editor.org/info/rfc7817>.

15.2. Informative References

[RFC2595] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, DOI 10.17487/RFC2595, June 1999, <http://www.rfc-editor.org/info/rfc2595>.

[RFC2979] Freed, N., "Behavior of and Requirements for Internet Firewalls", RFC 2979, DOI 10.17487/RFC2979, October 2000, <http://www.rfc-editor.org/info/rfc2979>.

[RFC3848] Newman, C., "ESMTP and LMTP Transmission Types Registration", RFC 3848, DOI 10.17487/RFC3848, July 2004, <http://www.rfc-editor.org/info/rfc3848>.

[RFC3887] Hansen, T., "Message Tracking Query Protocol", RFC 3887, DOI 10.17487/RFC3887, September 2004, <http://www.rfc-editor.org/info/rfc3887>.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, DOI 10.17487/RFC4346, April 2006, <http://www.rfc-editor.org/info/rfc4346>.

[RFC4422] Melnikov, A., Ed. and K. Zeilenga, Ed., "Simple Authentication and Security Layer (SASL)", RFC 4422, DOI 10.17487/RFC4422, June 2006, <http://www.rfc-editor.org/info/rfc4422>.

[RFC4954] Siemborski, R., Ed. and A. Melnikov, Ed., "SMTP Service Extension for Authentication", RFC 4954, DOI 10.17487/RFC4954, July 2007, <http://www.rfc-editor.org/info/rfc4954>.

[RFC5598] Crocker, D., "Internet Mail Architecture", RFC 5598, DOI 10.17487/RFC5598, July 2009, <http://www.rfc-editor.org/info/rfc5598>.

[RFC5804] Melnikov, A., Ed. and T. Martin, "A Protocol for Remotely Managing Sieve Scripts", RFC 5804, DOI 10.17487/RFC5804, July 2010, <http://www.rfc-editor.org/info/rfc5804>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <http://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, <http://www.rfc-editor.org/info/rfc6125>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <http://www.rfc-editor.org/info/rfc6335>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <http://www.rfc-editor.org/info/rfc6698>.

Appendix A. Design Considerations

This section is not normative.

The first version of this was written independently from draft-moore-email-tls-00.txt; subsequent versions merge ideas from both drafts.

One author of this document was also the author of RFC 2595 that became the standard for TLS usage with POP and IMAP, and the other author was perhaps the first to propose that idea. In hindsight both authors now believe that that approach was a mistake. At this point the authors believe that while anything that makes it easier to deploy TLS is good, the desirable end state is that these protocols always use TLS, leaving no need for a separate port for cleartext operation except to support legacy clients while they continue to be used. The separate port model for TLS is inherently simpler to implement, debug and deploy. It also enables a "generic TLS load-balancer" that accepts secure client connections for arbitrary foo-over-TLS protocols and forwards them to a server that may or may not support TLS. Such load-balancers cause many problems because they violate the end-to-end principle and the server loses the ability to log security-relevant information about the client unless the protocol is designed to forward that information (as this specification does for the cipher suite). However, they can result in TLS deployment where it would not otherwise happen which is a sufficiently important goal that it overrides the problems.

Although STARTTLS appears only slightly more complex than separate-port TLS, we again learned the lesson that complexity is the enemy of

security in the form of the STARTTLS command injection vulnerability (CERT vulnerability ID #555316). Although there's nothing inherently wrong with STARTTLS, the fact it resulted in a common implementation error (made independently by multiple implementers) suggests it is a less secure architecture than Implicit TLS.

Section 7 of RFC 2595 critiques the separate-port approach to TLS. The first bullet was a correct critique. There are proposals in the http community to address that, and use of SRV records as described in RFC 6186 resolves that critique for email. The second bullet is correct as well, but not very important because useful deployment of security layers other than TLS in email is small enough to be effectively irrelevant. The third bullet is incorrect because it misses the desirable option of "use and latch-on TLS if available". The fourth bullet may be correct, but is not a problem yet with current port consumption rates. The fundamental error was prioritizing a perceived better design based on a mostly valid critique over real-world deployability. But getting security and confidentiality facilities actually deployed is so important it should trump design purity considerations.

Port 465 is presently used for two purposes: for submissions by a large number of clients and service providers and for the "urd" protocol by one vendor. Actually documenting this current state is controversial as discussed in the IANA considerations section. However, there is no good alternative. Registering a new port for submissions when port 465 is widely used for that purpose already will just create interoperability problems. Registering a port that's only used if advertised by an SRV record (RFC 6186) would not create interoperability problems but would require all client and server deployments and software to change significantly which is contrary to the goal of promoting more TLS use. Encouraging use of STARTTLS on port 587 would not create interoperability problems, but is unlikely to have impact on current undocumented use of port 465 and makes the guidance in this document less consistent. The remaining option is to document the current state of the world and support future use of port 465 for submission as this increases consistency and ease-of-deployment for TLS email submission.

Appendix B. Change Log

Changes since draft-ietf-uta-email-deep-05:

- Clarify throughout that the confidentiality assurance level associated with a mail account is a minimum level; attempt to distinguish this from the current confidentiality level provided by a connection between client and server.

- Change naming for confidentiality assurance levels: instead of "high" or "no" confidence, assign numbers 1 and 0 to them respectively. This because it seems likely that in the not-too-distant future, what was defined in -05 as "high" confidence will be considered insufficient, and calling that "high" confidence will become misleading. For example, relying entirely on a list of trusted CAs to validate server certificates from arbitrary parties, appears to be less and less reliable in practice at thwarting MITM attacks.

- Clarify that if some services associated with a mail account don’t meet the minimium confidentiality assurance level assigned to that account, other services that do meet that minimum confidentiality assurance level may continue to be used.

- Clarify that successful negotiation of at least TLS version 1.1 is required as a condition of meeting confidentiality assurance level 1.

- Clarify that validation of a server certificate using either DANE or PKIX is sufficient to meet the certificate validation requirement of confidentiality assurance level 1.

- Clarify that minimum confidentiality assurance levels are separate from security directives, and that the requrements of both mechanisms must be met.

- Explicitly cite an example that a security directive of tls-version=1.2 won’t be saved if the currently negotiated tls-version is 1.1. (This example already appeared a bit later in the text, but for author KM it seemed to make the mechanism clearer to use this example earlier.)

- Clarify some protocol examples as to whether PKIX or DANE was used to verify a server’s certificate.

- Remove most references to DEEP as the conversion from DEEP to MUA-STS seemed incomplete, but kept the DEEP command for use in POP3 on the assumption that author CN wanted it that way.

- Removed most references to "latch" and derivative words.

- Added pkix+dane as a value for the tls-cert directive, to indicate (from a server) that both PKIX and DANE validation will be supported, or (from a client) that both PKIX and DANE were used to validate a certificate. Also clarified what each of any, pkix, dane, and pkix+dane mean when advertised by a server and in particular that tls-cert=any provides no assurance of future PKIX

verifiability in contrast to tls-cert=pkix or tls-cert=pkix+dane. It seemed important to support the ability to evolve to using multiple trust anchors for certificate validation, but also to allow servers to have the option to migrate from PKIX to DANE if that made sense for them. This change seemed less disruptive than either defining additional directives, or allowing multiple instances of the same directive with different values to appear in the same advertisement.

- Clarify interaction of this specification with anti-virus / anti-spam mechanisms.

Changes since draft-ietf-uta-email-deep-04:

- Swap sections 5.1 and 5.3 ("Email Security Tags" and "Server DEEP Status") as that order may aid understanding of the model. Also rewrote parts of these two sections to try to make the model clearer.
- Add text about versioning of security tags to make the model clearer.
- Add example of security tag upgrade.
- Convert remaining mention of TLS 1.0 to TLS 1.1.
- Change document title from DEEP to MUA STS to align with SMTP relay STS.
  - Slight updates to abstract and introductions.
  - Rename security latches/tags to security directives.
  - Rename server DEEP status to STS policy.
  - Change syntax to use directive-style HSTS syntax.
- Make HSTS reference normative.
- Remove SMTP DSN header as that belongs in SMTP relay STS document.

Changes since draft-ietf-uta-email-deep-03:

- Add more references to ietf-uta-email-tls-certs in implementation requirements section.

- Replace primary reference to RFC 6125 with ietf-uta-email-tls-certs, so move RFC 6125 to informative list for this specification.

Changes since draft-ietf-uta-email-deep-02:

- Make reference to design considerations explicit rather than "elsewhere in this document".
- Change provider requirement so SMTP submission services are separate from SMTP MTA services as opposed to the previous phrasing that required the servers be separate (which is too restrictive).
- Update DANE SMTP reference

Changes since draft-ietf-uta-email-deep-01:

- Change text in tls11 and tls12 registrations to clarify certificate rules, including additional PKIX and DANE references.
- Change from tls10 to tls11 (including reference) as the minimum.
- Fix typo in example 5.
- Remove open issues section; enough time has passed so not worth waiting for more input.

Changes since draft-ietf-uta-email-deep-00:

- Update and clarify abstract
- use term confidentiality instead of privacy in most cases.
- update open issues to request input for missing text.
- move certificate pinning sub-section to account setup section and attempt to define it more precisely.
- Add note about end-to-end encryption in AVAS section.
- swap order of DNSSEC and TLSA sub-sections.
- change meaning of ’tls10’ and ’tls12’ latches to require certificate validation.

- Replace cipher suite advice with reference to RFC 7525. Change examples to use TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as cipher suite.
- Add text to update IMAP, POP3 and Message Submission standards with newer TLS advice.
- Add clearer text in introduction that this does not cover SMTP relay.
- Update references to uta-tls-certs.
- Add paragraph to Implicit TLS for SMTP Submission section recommending that STARTTLS also be implemented.

Changes since draft-newman-email-deep-02:

- Changed "privacy assurance" to "confidentiality assurance"
- Changed "low privacy assurance" to "no confidentiality assurance"
- Attempt to improve definition of confidentiality assurance level.
- Add SHOULD indicate when MUA is showing list of mail accounts.
- Add SHOULD NOT latch tls10, tls12 tags until TLS negotiated.
- Removed sentence about deleting and re-creating the account in latch failure section.
- Remove use of word "fallback" with respect to TLS version negotiation.
- Added bullet about changes to Internet facing servers to MSP section.
- minor wording improvements based on feedback

Changes since -01:

- Updated abstract, introduction and document structure to focus more on mail user agent privacy assurance.
- Added email account privacy section, also moving section on account setup using SRV records to that section.
- Finished writing IANA considerations section

- Remove provisional concept and instead have server explicitly list security tags clients should latch.
- Added note that rules for the submissions port follow the same rules as those for the submit port.
- Reference and update advice in [RFC5068].
- Fixed typo in Client Certificate Authentication section.
- Removed tls-pfs security latch and all mention of perfect forward secrecy as it was controversial.
- Added reference to HSTS.

Changes since -00:

- Rewrote introduction to merge ideas from draft-moore-email-tls-00.
- Added Implicit TLS section, Account configuration section and IANA port registration updates based on draft-moore-email-tls-00.
- Add protocol details necessary to standardize implicit TLS for POP/IMAP/submission, using ideas from draft-melnikov-pop3-over-tls.
- Reduce initial set of security tags based on feedback.
- Add deep status concept to allow a window for software updates to be backed out before latches make that problematic, as well as to provide service providers with a mechanism they can use to assist customers in the event of a privacy failure.
- Add DNS SRV section from draft-moore-email-tls-00.
- Write most of the missing IANA considerations section.
- Rewrite most of implementation requirements section based more on draft-moore-email-tls-00. Remove new cipher requirements for now because those may be dealt with elsewhere.

Appendix C. Acknowledgements

Thanks to Ned Freed for discussion of the initial latch concepts in this document. Thanks to Alexey Melnikov for draft-melnikov-pop3-over-tls-02, which was the basis of the POP3 implicit TLS text. Thanks to Russ Housley, Alexey Melnikov and Dan Newman for review

feedback. Thanks to Paul Hoffman for interesting feedback in initial conversations about this idea.

Authors' Addresses

Keith Moore
Network Heretics
PO Box 1934
Knoxville, TN 37901
US

Email: moore@network-heretics.com

Chris Newman
Oracle
440 E. Huntington Dr., Suite 400
Arcadia, CA 91006
US

Email: chris.newman@oracle.com

Using TLS in Applications
Internet-Draft
Intended status: Standards Track
Expires: November 4, 2017

D. Margolis
M. Risher
Google, Inc
B. Ramakrishnan
Yahoo!, Inc
A. Brotman
Comcast, Inc
J. Jones
Microsoft, Inc
May 3, 2017

SMTP MTA Strict Transport Security (MTA-STS)
draft-ietf-uta-mta-sts-05

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 November 4, 2017.

Copyright Notice

Table of Contents

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 specifying:

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

1.1. Terminology

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

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

- MTA-STS Policy: A commitment by the Policy Domain to support PKIX authenticated TLS for the specified MX hosts.
- 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 ordinarly be "example.com", but this may be overriden by explicit routing rules (as described in Section 3.4, "Policy Selection for Smart Hosts and Subdomains").

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-resistent encrypted tarnsmission. 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 8, "Security Considerations".

In addition, MTA-STS provides an optional report-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 upgrade their 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 "report-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:

- "v": (plain-text, required). Currently only "STSv1" is supported.
- "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 [RFC5234], is as follows:

```
sts-text-record = sts-version *WSP field-delim *WSP sts-id
                  [field-delim [sts-extensions]]

field-delim     = %x3B                                ; ";"

sts-version     = %x76 *WSP "=" *WSP %x53 %x54        ; "v=STSv1"
                  %x53 %x76 %x31

sts-id          = %x69 %x64 *WSP "="
                  *WSP 1*32(ALPHA / DIGIT)            ; "id="

sts-extensions  = sts-extension *(field-delim sts-extension)
                  [field-delim]                       ; extension fields

sts-extension   = sts-ext-name *WSP "=" *WSP sts-ext-value

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

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

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, senders MUST assume the recipient domain does not implement MTA-STS and skip the remaining steps of policy discovery.

3.2. MTA-STS Policies

The policy itself is a JSON [RFC7159] object served via the HTTPS GET method from the fixed [RFC5785] "well-known" path of ".well-known/mta-sts.json" served by the "mta-sts" host at the Policy Domain. Thus for "example.com" the path is "https://mta-sts.example.com/.well-known/mta-sts.json".

This JSON object contains the following key/value pairs:

- "version": (plain-text, required). Currently only "STSv1" is supported.
- "mode": (plain-text, required). Either "enforce" or "report", indicating the expected behavior of a sending MTA in the case of a policy validation failure.
- "max_age": Max lifetime of the policy (plain-text non-negative integer seconds, required). 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.

- "mx": MX identity patterns (list of plain-text strings, required). One or more patterns matching a Common Name ([RFC6125]) or Subject Alternative Name ([RFC5280]) DNS-ID present in the X.509 certificate presented by any MX receiving mail for this domain. For example, "["mail.example.com", ".example.net"]" indicates that mail for this domain might be handled by any MX with a certificate valid for a host at "mail.example.com" or "example.net". Valid patterns can be either fully specified names ("example.com") or suffixes (".example.net") matching the right-hand parts of a server's identity; the latter case are distinguished by a leading period. In the case of Internationalized Domain Names ([RFC5891]), the MX MUST specify the Punycode-encoded A-label [RFC3492] and not the Unicode-encoded U-label. The full semantics of certificate validation are described in Section 4.1, "MX Certificate Validation."

An example JSON policy is as below:

```
{
  "version": "STSv1",
  "mode": "enforce",
  "mx": [".mail.example.com"],
  "max_age": 123456
}
```

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 and valid JSON for policy files) and implementing a superset of this specification, in which case unknown fields SHALL be ignored.

3.3. HTTPS Policy Fetching

When fetching a new policy or updating a policy, the HTTPS endpoint MUST present a X.509 certificate which is valid for the "mta-sts" host (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.

The certificate is valid for the "mta-sts" host with respect to the rules described in [RFC6125], with the following application-specific considerations:

- Matching is performed only against the DNS-ID and CN-ID identifiers.

- 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) [RFC2560], certificate revocation lists (CRLs), or some other mechanism.

HTTP 3xx redirects MUST NOT be followed.

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, we suggest implementions may 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. Senders who implement TLSRPT (TODO: add ref) should, however, report this failure to the recipient domain if the domain implements TLSRPT as well.

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.

3.4. Policy Selection for Smart Hosts and Subdomains

When sending mail via a "smart host"--an intermediate SMTP relay rather than the message recipient's server--compliant senders MUST treat the smart host domain as the policy domain for the purposes of policy discovery and application.

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 validate:

1. That the recipient MX supports STARTTLS and offers a valid PKIX-based TLS certificate.

2. That at least one of the policy's "mx" patterns matches at least one of the identities presented in the MX's X.509 certificate, as described in "MX Certificate Validation".

This section does not dictate the behavior of sending MTAs when policies fail to validate; in particular, validation failures of policies which specify "report" mode MUST NOT be interpreted as delivery failures, as described in Section 5, "Policy Application".

4.1. MX Certificate Validation

The certificate presented by the receiving MX MUST chain to a root CA that is trusted by the sending MTA and be non-expired. The certificate MUST have a CN-ID ([RFC6125]) or SAN ([RFC5280]) with a DNS-ID matching the "mx" pattern. The MX's certificate MAY also be checked for revocation via OCSP [RFC2560], certificate revocation lists (CRLs), or some other mechanism.

Because the "mx" patterns are not hostnames, however, matching is not identical to other common cases of X.509 certificate authentication (as described, for example, in [RFC6125]). Consider the example policy given above, with an "mx" pattern containing ".example.net". In this case, if the MX server's X.509 certificate contains a SAN matching "*.example.net", we are required to implement "wildcard-to-wildcard" matching.

To simplify this case, we impose the following constraints on wildcard certificates, identical to those in [RFC7672] section 3.2.3 and [@!RFC6125 section 6.4.3: wildcards are valid in DNS-IDs or CN-IDs, but must be the entire first label of the identifier (that is, "*.example.com", not "mail*.example.com"). Senders who are comparing a "suffix" MX pattern with a wildcard identifier should thus strip the wildcard and ensure that the two sides match label-by-label, until all labels of the shorter side (if unequal length) are consumed.

A simple pseudocode implementation of this algorithm is presented in the Appendix.

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. "report": In this mode, sending MTAs merely send a report (as described in the TLSRPT specification (TODO: add ref)) indicating policy application failures.

2. "enforce": In this mode, sending MTAs MUST NOT deliver the message to hosts which fail MX matching or certificate validation.

When a message fails to deliver due to an "enforce" policy, a compliant MTA MUST NOT permanently fail to deliver messages before checking 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.

Finally, in both "enforce" and "report" modes, failures to deliver in compliance with the applied policy result in failure reports to the policy domain, as described in the TLSRPT specification (TODO: add ref).

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, enforcing STARTTLS and, assuming a policy is present, PKIX certificate validation as described in Section 4.1, "MX Certificate Validation."

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, senders SHOULD apply existing rules for the case of temporary message delivery failures (as discussed in [RFC5321] section 4.5.4.1).

6. Operational Considerations

6.1. 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 thus 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.

7. IANA Considerations

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

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

8. Security Considerations

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

- 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.

- 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.

8.1. 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.

8.2. 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 victim 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, we strongly recommend implementors to prefer policy "max_age" values to be as long as is practical.

Because this attack is also possible upon refresh of a cached policy, we suggest implementors do not wait until a cached policy has expired before checking for an update; if senders attempt to refresh the cache regularly (for instance, by checking their cached version string against the TXT record on each successful send, or in a background task that runs daily or weekly), an attacker would have to foil policy discovery consistently over the lifetime of a cached policy to prevent a successful refresh.

Resistence 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.

8.3. Denial of Service

We additionally consider the Denial of Service risk posed by an attacker who can modify the DNS records for a victim 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.

8.4. 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.

9. Contributors

Nicolas Lidzborski Google, Inc nlidz (at) google (dot com)

Wei Chuang Google, Inc weihaw (at) google (dot com)

Brandon Long Google, Inc blong (at) google (dot com)

Franck Martin LinkedIn, Inc fmartin (at) linkedin (dot com)

Klaus Umbach 1&1 Mail & Media Development & Technology GmbH
klaus.umbach (at) 1und1 (dot de)

Markus Laber 1&1 Mail & Media Development & Technology GmbH
markus.laber (at) 1und1 (dot de)

10. Appendix 1: 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 JSON served as the response body at [1]

```
{
  "version": "STSv1",
  "mode": "report",
  "mx": ["mx1.example.com", "mx2.example.com"],
  "max_age": 12345678
}
```

11. Appendix 2: 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 implementors 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, policy) {
  // Assume a handy function to return CN and DNS-ID SANs.
  for san in getDnsIdSansAndCnFromCert(connection) {
    for mx in policy.mx {
      // Return if the server certificate from "connection" matches the "mx" hos
t.
      if san[0] == '*' {
        // Invalid wildcard!
        if san[1] != '.' return false
        san = san[1:]
      }
      if san[0] == '.' && HasSuffix(mx, san) {
        return true
      }
      if mx[0] == '.' && HasSuffix(san, mx) {
        return true
      }
      if mx == san {
        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 !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).
}
```

12. References

12.1. Normative References

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

[RFC2560] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, DOI 10.17487/RFC2560, June 1999, <http://www.rfc-editor.org/info/rfc2560>.

[RFC3207] Hoffman, P., "SMTP Service Extension for Secure SMTP over Transport Layer Security", RFC 3207, DOI 10.17487/RFC3207, February 2002, <http://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, <http://www.rfc-editor.org/info/rfc3492>.

[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, <http://www.rfc-editor.org/info/rfc4033>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[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, <http://www.rfc-editor.org/info/rfc5280>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <http://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, <http://www.rfc-editor.org/info/rfc5785>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <http://www.rfc-editor.org/info/rfc5891>.

[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, <http://www.rfc-editor.org/info/rfc6125>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[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, <http://www.rfc-editor.org/info/rfc7672>.

12.2. URIs

[1] https://mta-sts.example.com/.well-known/mta-sts.json:

Authors' Addresses

Daniel Margolis
Google, Inc

Email: dmargolis (at) google.com

Mark Risher
Google, Inc

Email: risher (at) google (dot com)

Binu Ramakrishnan
Yahoo!, Inc

Email: rbinu (at) yahoo-inc (dot com)

Alexander Brotman
Comcast, Inc

Email: alex_brotman (at) comcast.com

Janet Jones
Microsoft, Inc

Email: janet.jones (at) microsoft (dot com)

Using TLS in Applications
Internet-Draft
Intended status: Standards Track
Expires: November 4, 2017

D. Margolis
Google, Inc
A. Brotman
Comcast, Inc
B. Ramakrishnan
Yahoo!, Inc
J. Jones
Microsoft, Inc
M. Risher
Google, Inc
May 3, 2017

SMTP TLS Reporting
draft-ietf-uta-smtp-tlsrpt-05

Abstract

A number of protocols exist for establishing encrypted channels between SMTP Mail Transfer Agents, including STARTTLS [RFC3207], DANE [RFC6698], and MTA-STS (TODO: Add ref). These protocols can fail due to misconfiguration or active attack, leading to undelivered messages or delivery over unencrypted or unauthenticated channels. This document describes a reporting mechanism and format by which sending systems can share statistics and specific information about potential failures with recipient domains. Recipient domains can then use this information to both detect potential attackers and diagnose unintentional misconfigurations.

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 November 4, 2017.

Table of Contents

1. Introduction

The STARTTLS extension to SMTP [RFC3207] allows SMTP clients and hosts to establish secure SMTP sessions over TLS. The protocol design is based on "Opportunistic Security" (OS) [RFC7435], which maintains interoperability with clients that do not support STARTTLS but means that any attacker who can delete parts of the SMTP session (such as the "250 STARTTLS" response) or redirect the entire SMTP session (perhaps by overwriting the resolved MX record of the delivery domain) can perform a downgrade or interception attack.

Because such "downgrade attacks" are not necessarily apparent to the receiving MTA, this document defines a mechanism for sending domains to report on failures at multiple stages of the MTA-to-MTA conversation.

Recipient domains may also use the mechanisms defined by MTA-STS (TODO: Add ref) or DANE [RFC6698] to publish additional encryption and authentication requirements; this document defines a mechanism for sending domains that are compatible with MTA-STS or DANE to share success and failure statistics with recipient domains.

Specifically, this document defines a reporting schema that covers failures in routing, STARTTLS negotiation, and both DANE [RFC6698] and MTA-STS (TODO: Add ref) policy validation errors, and a standard TXT record that recipient domains can use to indicate where reports in this format should be sent.

This document is intended as a companion to the specification for SMTP MTA Strict Transport Security (MTA-STS, TODO: Add ref).

1.1. Terminology

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

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

- MTA-STS Policy: A definition of the expected TLS availability, behavior, and desired actions for a given domain when a sending MTA encounters problems in negotiating a secure channel. MTA-STS is defined in [TODO]

- DANE Policy: A mechanism by which administrators can supply a record that can be used to validate the certificate presented by an MTA. DANE is defined in [RFC6698].

- TLSRPT Policy: A policy specifying the endpoint to which sending MTAs should deliver reports.

- Policy Domain: The domain against which an MTA-STS or DANE Policy is defined.

- Sending MTA: The MTA initiating the delivery of an email message.

2. Related Technologies

- This document is intended as a companion to the specification for SMTP MTA Strict Transport Security (MTA-STS, TODO: Add ref).

- SMTP-TLSRPT defines a mechanism for sending domains that are compatible with MTA-STS or DANE to share success and failure statistics with recipient domains. DANE is defined in [RFC6698] and MTA-STS is defined in [TODO]

3. Reporting Policy

A domain publishes a record to its DNS indicating that it wishes to receive reports. These SMTP TLSRPT policies are distributed via DNS from the Policy Domain's zone, as TXT records (similar to DMARC policies) under the name "_smtp-tlsrpt". For example, for the Policy Domain "example.com", the recipient's TLSRPT policy can be retrieved from "_smtp-tlsrpt.example.com".

Policies consist of the following directives:

- "v": This value MUST be equal to "TLSRPTv1".

- "rua": A URI specifying the endpoint to which aggregate information about policy failures should be sent (see Section 4, "Reporting Schema", for more information). Two URI schemes are supported: "mailto" and "https".

- In the case of "https", reports should be submitted via POST ([RFC2818]) to the specified URI.

- In the case of "mailto", reports should be submitted to the specified email address ([RFC6068]). When sending failure reports via SMTP, sending MTAs MUST deliver reports despite any TLS-related failures. This may mean that the reports are delivered in the clear.

The formal definition of the "_smtp-tlsrpt" TXT record, defined using [RFC5234], is as follows:

```
 tlsrpt-record     = tlsrpt-version *WSP field-delim *WSP tlsrpt-rua
                     [field-delim [tlsrpt-extensions]]

 field-delim       = %x3B                                    ; ";"

 tlsrpt-version    = %x76 *WSP "=" *WSP %x54 %x4C %x53 %x52
                     %x50 %x54 %x76 %x31              ; "v=TSRPTv1"

 tlsrpt-rua        = %x72 %x75 %x61 *WSP "=" *WSP tlsrpt-uri ; "rua=..."

 tlsrpt-uri        = URI
                   ; "URI" is imported from [@!RFC3986]; commas (ASCII
                   ; 0x2C) and exclamation points (ASCII 0x21)
                   ; MUST be encoded; the numeric portion MUST fit
                   ; within an unsigned 64-bit integer

 tlsrpt-extensions = tlsrpt-extension *(field-delim tlsrpt-extension)
                     [field-delim]
                   ; extension fields

 tlsrpt-extension  = tlsrpt-ext-name *WSP "=" *WSP tlsrpt-ext-value

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

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

If multiple TXT records for "_smtp-tlsrpt" are returned by the resolver, records which do not begin with "v=TLSRPTv1;" are discarded. If the number of resulting records is not one, senders MUST assume the recipient domain does not implement TLSRPT. Parsers MUST accept TXT records which are syntactically valid (i.e. valid key-value pairs seprated by semi-colons) and implementing a superset of this specification, in which case unknown fields SHALL be ignored.

3.1. Example Reporting Policy

3.1.1. Report using MAILTO

```
_smtp-tlsrpt.example.com. IN TXT \
    "v=TLSRPTv1;rua=mailto:reports@example.com"
```

3.1.2. Report using HTTPS

```
_smtp-tlsrpt.example.com. IN TXT \
    "v=TLSRPTv1; \
    rua=https://reporting.example.com/v1/tlsrpt"
```

4. Reporting Schema

The report is composed as a plain text file encoded in the JSON format ([RFC7159]).

Aggregate reports contain the following fields:

- Report metadata:
  - The organization responsible for the report
  - Contact information for one or more responsible parties for the contents of the report
  - A unique identifier for the report
  - The reporting date range for the report
- Policy, consisting of:
  - One of the following policy types: (1) The MTA-STS policy applied (as a string) (2) The DANE TLSA record applied (as a string, with each RR entry of the RRset listed and separated by a semicolon) (3) The literal string "no-policy-found", if neither a TLSA nor MTA-STS policy could be found.
  - The domain for which the policy is applied
  - The MX host
  - An identifier for the policy (where applicable)
- Aggregate counts, comprising result type, sending MTA IP, receiving MTA hostname, session count, and an optional additional information field containing a URI for recipients to review further information on a failure type.

Note that the failure types are non-exclusive; an aggregate report may contain overlapping "counts" of failure types when a single send attempt encountered multiple errors.

4.1. Report Time-frame

The report SHOULD cover a full day, from 0000-2400 UTC. This should allow for easier correlation of failure events.

4.2. Delivery Summary

4.2.1. Success Count

- "success-count": This indicates that the sending MTA was able to successfully negotiate a policy-compliant TLS connection, and serves to provide a "heartbeat" to receiving domains that reporting is functional and tabulating correctly. This field contains an aggregate count of successful connections for the reporting system.

4.2.2. Failure Count

- "failure-count": This indicates that the sending MTA was unable to successfully establish a connection with the receiving platform. Section 4.3, "Result Types", will elaborate on the failed negotiation attempts. This field contains an aggregate count of failed connections.

4.3. Result Types

The list of result types will start with the minimal set below, and is expected to grow over time based on real-world experience. The initial set is:

4.3.1. Negotiation Failures

- "starttls-not-supported": This indicates that the recipient MX did not support STARTTLS.

- "certificate-host-mismatch": This indicates that the certificate presented did not adhere to the constraints specified in the MTA-STS or DANE policy, e.g. if the MX does not match any identities listed in the Subject Alternate Name (SAN) [RFC5280].

- "certificate-expired": This indicates that the certificate has expired.

- "certificate-not-trusted": This a label that covers multiple certificate related failures that include, but not limited to errors such as untrusted/unknown CAs, certificate name constraints, certificate chain errors etc. When using this

declaration, the reporting MTA SHOULD utilize the "failure-reason" to provide more information to the receiving entity.

- "validation-failure": This indicates a general failure for a reason not matching a category above. When using this declaration, the reporting MTA SHOULD utilize the "failure-reason" to provide more information to the receiving entity.

4.3.2. Policy Failures

4.3.2.1. DANE-specific Policy Failures

- "tlsa-invalid": This indicates a validation error in the TLSA record associated with a DANE policy. None of the records in the RRset were found to be valid.

- "dnssec-invalid": This would indicate that no valid records were returned from the recursive resolver. The request returned with SERVFAIL for the requested TLSA record.

4.3.2.2. MTA-STS-specific Policy Failures

- "sts-invalid": This indicates a validation error for the overall MTA-STS policy.

- "webpki-invalid": This indicates that the MTA-STS policy could not be authenticated using PKIX validation.

4.3.3. General Failures

When a negotiation failure can not be categorized into one of the "Negotiation Failures" stated above, the reporter SHOULD use the "validation-failure" category. As TLS grows and becomes more complex, new mechanisms may not be easily categorized. This allows for a generic feedback category. When this category is used, the reporter SHOULD also use the "failure-reason-code" to give some feedback to the receiving entity. This is intended to be a short text field, and the contents of the field should be an error code or error text, such as "X509_V_ERR_UNHANDLED_CRITICAL_CRL_EXTENSION".

4.3.4. Transient Failures

Transient errors due to too-busy network, TCP timeouts, etc. are not required to be reported.

5. Report Delivery

Reports can be delivered either as an email message via SMTP or via HTTP POST.

5.1. Report Filename

The filename is typically constructed using the following ABNF:

```
filename = sender "!" policy-domain "!" begin-timestamp
           "!" end-timestamp [ "!" unique-id ] "." extension

unique-id = 1*(ALPHA / DIGIT)

sender = domain        ; imported from [@!RFC5322]

policy-domain   = domain

begin-timestamp = 1*DIGIT
                  ; seconds since 00:00:00 UTC January 1, 1970
                  ; indicating start of the time range contained
                  ; in the report

end-timestamp = 1*DIGIT
                  ; seconds since 00:00:00 UTC January 1, 1970
                  ; indicating end of the time range contained
                  ; in the report

extension = "json" / "json.gz"
```

The extension MUST be "json" for a plain JSON file, or "json.gz" for a JSON file compressed using GZIP.

"unique-id" allows an optional unique ID generated by the Sending MTA to distinguish among multiple reports generated simultaneously by different sources within the same Policy Domain. For example, this is a possible filename for the gzip file of a report to the Policy Domain "example.net" from the Sending MTA "mail.sender.example.com":

‘mail.sender.example.com!example.net!1470013207!1470186007!001.json.gz‘

5.2. Compression

The report SHOULD be subjected to GZIP compression for both email and HTTPS transport. Declining to apply compression can cause the report to be too large for a receiver to process (a commonly observed receiver limit is ten megabytes); compressing the file increases the chances of acceptance of the report at some compute cost.

5.3. Email Transport

The report MAY be delivered by email. No specific MIME message structure is required. It is presumed that the aggregate reporting address will be equipped to extract MIME parts with the prescribed media type and filename and ignore the rest.

If compressed, the report should use the media type "application/gzip" if compressed (see [RFC6713]), and "application/json" otherwise.

The [RFC5322].Subject field for individual report submissions SHOULD conform to the following ABNF:

```
  tlsrpt-subject = %x52.65.70.6f.72.74 1*FWS       ; "Report"
                   %x44.6f.6d.61.69.6e.3a 1*FWS    ; "Domain:"
                   domain-name 1*FWS               ; from RFC 6376
                   %x53.75.62.6d.69.74.74.65.72.3a ; "Submitter:"
                   1*FWS domain-name 1*FWS
                   %x52.65.70.6f.72.74.2d.49.44.3a ; "Report-ID:"
                   msg-id                          ; from RFC 5322
```

The first domain-name indicates the DNS domain name about which the report was generated. The second domain-name indicates the DNS domain name representing the Sending MTA generating the report. The purpose of the Report-ID: portion of the field is to enable the Policy Domain to identify and ignore duplicate reports that might be sent by a Sending MTA.

For instance, this is a possible Subject field for a report to the Policy Domain "example.net" from the Sending MTA "mail.sender.example.com". It is line-wrapped as allowed by [RFC5322]:

```
          Subject: Report Domain: example.net
              Submitter: mail.sender.example.com
              Report-ID: <735ff.e317+bf22029@mailexample.net>
```

Note that, when sending failure reports via SMTP, sending MTAs MUST NOT honor MTA-STS or DANE TLSA failures.

5.4. HTTPS Transport

The report MAY be delivered by POST to HTTPS. If compressed, the report should use the media type "application/gzip" (see [RFC6713]), and "application/json" otherwise.

5.5. Delivery Retry

In the event of a delivery failure, regardless of the delivery method, a sender SHOULD attempt redelivery for up to 24hrs after the initial attempt. As previously stated the reports are optional, so while it is ideal to attempt redelivery, it is not required. If multiple retries are attempted, they should be on a logarithmic scale.

6. IANA Considerations

There are no IANA considerations at this time.

7. Security Considerations

SMTP TLS Reporting provides transparency into misconfigurations or attempts to intercept or tamper with mail between hosts who support STARTTLS. There are several security risks presented by the existence of this reporting channel:

- Flooding of the Aggregate report URI (rua) endpoint: An attacker could flood the endpoint and prevent the receiving domain from accepting additional reports. This type of Denial-of-Service attack would limit visibility into STARTTLS failures, leaving the receiving domain blind to an ongoing attack.

- Untrusted content: An attacker could inject malicious code into the report, opening a vulnerability in the receiving domain. Implementers are advised to take precautions against evaluating the contents of the report.

- Report snooping: An attacker could create a bogus TLSRPT record to receive statistics about a domain the attacker does not own. Since an attacker able to poison DNS is already able to receive counts of SMTP connections (and, absent DANE or MTA-STS policies, actual SMTP message payloads), this does not present a significant new vulnerability.

- Reports as DDoS: TLSRPT allows specifying destinations for the reports that are outside the authority of the Policy Domain, which allows domains to delegate processing of reports to a partner organization. However, an attacker who controls the Policy Domain DNS could also use this mechanism to direct the reports to an unwitting victim, flooding that victim with excessive reports. DMARC [RFC7489] defines an elegant solution for verifying delegation; however, since the attacker had less ability to generate large reports than with DMARC failures, and since the

reports are generated by the sending MTA, such a delegation mechanism is left for a future version of this specification.

8. Appendix 1: Example Reporting Policy

8.1. Report using MAILTO

```
_smtp-tlsrpt.mail.example.com. IN TXT \
        "v=TLSRPTv1;rua=mailto:reports@example.com"
```

8.2. Report using HTTPS

```
_smtp-tlsrpt.mail.example.com. IN TXT \
        "v=TLSRPTv1; \
        rua=https://reporting.example.com/v1/tlsrpt"
```

9. Appendix 2: JSON Report Schema

The JSON schema is derived from the HPKP JSON schema [RFC7469] (cf. Section 3)

```
{
  "organization-name": organization-name,
  "date-range": {
    "start-datetime": date-time,
    "end-datetime": date-time
  },
  "contact-info": email-address,
  "report-id": report-id,
  "policy": {
    "policy-type": policy-type,
    "policy-string": policy-string,
    "policy-domain": domain,
    "mx-host": mx-host-pattern
  },
  "summary": {
    "success-aggregate": total-successful-session-count,
    "failure-aggregate:" total-failure-session-count
  }
  "failure-details": [
    {
      "result-type": result-type,
      "sending-mta-ip": ip-address,
      "receiving-mx-hostname": receiving-mx-hostname,
      "receiving-mx-helo": receiving-mx-helo,
      "session-count": failed-session-count,
      "additional-information": additional-info-uri,
      "failure-reason-code": "Text body"
    }
  ]
}
```

Figure: JSON Report Format

- "organization-name": The name of the organization responsible for the report. It is provided as a string.

- "date-time": The date-time indicates the start- and end-times for the report range. It is provided as a string formatted according to Section 5.6, "Internet Date/Time Format", of [RFC3339]. The report should be for a full UTC day, 0000-2400.

- "email-address": The contact information for a responsible party of the report. It is provided as a string formatted according to Section 3.4.1, "Addr-Spec", of [RFC5322].

- "report-id": A unique identifier for the report. Report authors may use whatever scheme they prefer to generate a unique identifier. It is provided as a string.

- "policy-type": The type of policy that was applied by the sending domain. Presently, the only three valid choices are "tlsa", "sts", and the literal string "no-policy-found". It is provided as a string.

- "policy-string": The JSON string serialization ([RFC7159] section 7) of the policy, whether TLSA record ([RFC6698] section 2.3) or MTA-STS policy.

- "domain": The Policy Domain is the domain against which the MTA-STS or DANE policy is defined.

- "mx-host-pattern": The pattern of MX hostnames from the applied policy. It is provided as a string, and is interpreted in the same manner as the "Checking of Wildcard Certificates" rules in Section 6.4.3 of [RFC6125].

- "result-type": A value from Section 4.3, "Result Types", above.

- "ip-address": The IP address of the sending MTA that attempted the STARTTLS connection. It is provided as a string representation of an IPv4 or IPv6 address in dot-decimal or colon-hexadecimal notation.

- "receiving-mx-hostname": The hostname of the receiving MTA MX record with which the sending MTA attempted to negotiate a STARTTLS connection.

- "receiving-mx-helo": (optional) The HELO or EHLO string from the banner announced during the reported session.

- "success-aggregate": The aggregate number (integer) of successfully negotiated TLS-enabled connections to the receiving site.

- "failure-aggregate": The aggregate number (integer) of failures to negotiate an TLS-enabled connection to the receiving site.

- "session-count": The number of (attempted) sessions that match the relevant "result-type" for this section.

- "additional-info-uri": An optional URI pointing to additional information around the relevant "result-type". For example, this URI might host the complete certificate chain presented during an attempted STARTTLS session.

- "failure-reason-code": A text field to include an TLS-related error code or error message.

10. Appendix 3: Example JSON Report

```
{
  "organization-name": "Company-X",
  "date-range": {
    "start-datetime": "2016-04-01T00:00:00Z",
    "end-datetime": "2016-04-01T23:59:59Z"
  },
  "contact-info": "sts-reporting@company-x.com",
  "report-id": "5065427c-23d3-47ca-b6e0-946ea0e8c4be",
  "policy": {
    "policy-type": "sts",
    "policy-string": "{ \"version\": \"STSv1\",\"mode\": \"report\", \"mx\": [\"
*.mail.company-y.com\"], \"max_age\": 86400 }",
    "policy-domain": "company-y.com",
    "mx-host": "*.mail.company-y.com"
  },
  "summary": {
    "success-aggregate": 5326,
    "failure-aggregate": 303
  }
  "failure-details": [{
    "result-type": "certificate-expired",
    "sending-mta-ip": "98.136.216.25",
    "receiving-mx-hostname": "mx1.mail.company-y.com",
    "session-count": 100
  }, {
    "result-type": "starttls-not-supported",
    "sending-mta-ip": "98.22.33.99",
    "receiving-mx-hostname": "mx2.mail.company-y.com",
    "session-count": 200,
    "additional-information": "hxxps://reports.company-x.com/
      report_info?id=5065427c-23d3#StarttlsNotSupported"
  }, {
    "result-type: "validation-failure",
    "sending-mta-ip": "47.97.15.2",
    "receiving-mx-hostname: "mx-backup.mail.company-y.com",
    "session-count": 3,
    "failure-error-code": "X509_V_ERR_PROXY_PATH_LENGTH_EXCEEDED"
  }]
}
```

Figure: Example JSON report for a messages from Company-X to Company-Y, where 100 sessions were attempted to Company Y servers with an expired certificate and 200 sessions were attempted to Company Y servers that did not successfully respond to the "STARTTLS" command. Additionally 3 sessions failed due to "X509_V_ERR_PROXY_PATH_LENGTH_EXCEEDED".

11. Normative References

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

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://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, <http://www.rfc-editor.org/info/rfc3207>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <http://www.rfc-editor.org/info/rfc3339>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <http://www.rfc-editor.org/info/rfc5322>.

[RFC6068] Duerst, M., Masinter, L., and J. Zawinski, "The 'mailto' URI Scheme", RFC 6068, DOI 10.17487/RFC6068, October 2010, <http://www.rfc-editor.org/info/rfc6068>.

[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, <http://www.rfc-editor.org/info/rfc6125>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <http://www.rfc-editor.org/info/rfc6698>.

[RFC6713] Levine, J., "The 'application/zlib' and 'application/gzip' Media Types", RFC 6713, DOI 10.17487/RFC6713, August 2012, <http://www.rfc-editor.org/info/rfc6713>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <http://www.rfc-editor.org/info/rfc7435>.

[RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <http://www.rfc-editor.org/info/rfc7469>.

[RFC7489] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based Message Authentication, Reporting, and Conformance (DMARC)", RFC 7489, DOI 10.17487/RFC7489, March 2015, <http://www.rfc-editor.org/info/rfc7489>.

Authors' Addresses

Daniel Margolis
Google, Inc

Email: dmargolis (at) google.com

Alexander Brotman
Comcast, Inc

Email: alex_brotman (at) comcast.com

Binu Ramakrishnan
Yahoo!, Inc

Email: rbinu (at) yahoo-inc (dot com)

Janet Jones
Microsoft, Inc

Email: janet.jones (at) microsoft (dot com)

Mark Risher
Google, Inc

Email: risher (at) google (dot com)