

Using TLS in Applications
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

D. Margolis
M. Risher
N. Lidzborski
W. Chuang
B. Long
Google, Inc
B. Ramakrishnan
Yahoo!, Inc
A. Brotman
Comcast, Inc
J. Jones
Microsoft, Inc
F. Martin
LinkedIn
K. Umbach
M. Laber
1&1 Mail & Media Development & Technology GmbH
July 8, 2016

SMTP MTA Strict Transport Security
draft-ietf-uta-mta-sts-01

Abstract

SMTP MTA-STs is a mechanism enabling mail service providers to declare their ability to receive TLS-secured connections, to declare particular methods for certificate validation, and to request that sending SMTP servers report upon and/or refuse to deliver messages that cannot be delivered securely.

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 January 9, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
1.1.	Terminology	3
2.	Related Technologies	4
2.1.	Differences from DANE	4
2.1.1.	Advantages of SMTP MTA-STS when compared to DANE	4
2.1.2.	Advantages of DANE when compared to SMTP MTA-STS	5
3.	Policy Semantics	5
3.1.	Formal Definition	6
3.1.1.	TXT Record	6
3.1.2.	SMTP MTA-STS Policy	6
3.2.	Policy Expirations	7
3.2.1.	Policy Updates	8
3.3.	Policy Discovery & Authentication	8
3.4.	Policy Validation	9
3.5.	Policy Application	9
4.	Failure Reporting	10
5.	IANA Considerations	10
6.	Security Considerations	10
7.	Future Work	11
8.	Appendix 1: Validation Pseudocode	12
9.	Appendix 2: Domain Owner STS example record	12
9.1.	Example 1	12
10.	Appendix 3: DEEP Registration Elements	13
11.	Normative References	15
	Authors' Addresses	16

[1. Introduction](#)

The STARTTLS extension to SMTP [[RFC3207](#)] allows SMTP clients and hosts to establish secure SMTP sessions over TLS. In its current form, however, it fails to provide (a) message confidentiality --

because opportunistic STARTTLS is subject to downgrade attacks -- and (b) server authenticity -- because the trust relationship from email domain to MTA server identity is not cryptographically validated.

While such `_opportunistic_` encryption protocols provide 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 such a downgrade or interception attack.

This document defines a mechanism for recipient domains to publish policies specifying:

- o whether MTAs sending mail to this domain can expect TLS support
- o how MTAs can validate the TLS server certificate presented during mail delivery
- o the expected identity of MXs that handle mail for this domain
- o what an implementing sender should do with messages when TLS cannot be successfully negotiated

The mechanism described is separated into four logical components:

1. policy semantics: whether senders can expect a server for the recipient domain to support TLS encryption and how to validate the TLS certificate presented
2. policy discovery & authentication: how to discover a domain's published STS policy and determine the authenticity of that policy
3. failure report format: a mechanism for informing recipient domains about aggregate failure statistics
4. failure handling: what sending MTAs should do in the case of policy failures

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:

- o STS Policy: A definition of the expected TLS availability and behavior, as well as the desired actions for a given domain when a sending MTA encounters different results.
- o Policy Domain: The domain against which an STS Policy is defined.
- o Policy Authentication: Authentication of the STS policy retrieved for a recipient domain by the sender.

2. Related Technologies

The DANE TLSA record [[RFC7672](#)] is similar, in that DANE is also designed to upgrade opportunistic encryption into required encryption. DANE requires DNSSEC [[RFC4033](#)] for the secure delivery of policies; the mechanism described here presents a variant for systems not yet supporting DNSSEC.

2.1. Differences from DANE

The primary difference between the mechanism described here and DANE is that DANE requires the use of DNSSEC to authenticate DANE TLSA records, whereas SMTP STS relies on the certificate authority (CA) system to avoid interception. (For a thorough discussion of this trade-off, see the section `_Security_` `_Considerations_`.)

In addition, SMTP MTA-STS introduces a mechanism for failure reporting and a report-only mode, enabling offline ("report-only") deployments and auditing for compliance.

2.1.1. Advantages of SMTP MTA-STS when compared to DANE

SMTP MTA-STS offers the following advantages compared to DANE:

- o Infrastructure: In comparison to DANE, this proposal does not require DNSSEC be deployed on either the sending or receiving domain. In addition, the reporting feature of SMTP MTA-STS can be deployed to perform offline analysis of STARTTLS failures, enabling mail providers to gain insight into the security of their SMTP connections without the need to modify MTA codebases directly.
- o Offline or report-only usage: DANE does not provide a reporting mechanism and does not have a concept of "report-only" for failures; as a result, a service provider cannot receive metrics on TLS acceptability without asking senders to enforce a given policy; similarly, senders who cannot enforce DANE constraints at send-time have no mechanism to provide recipients such metrics

from an offline (and potentially easier-to-deploy) logs-analysis batch process.

2.1.2. Advantages of DANE when compared to SMTP MTA-STS

- o Infrastructure: DANE may be easier for some providers to deploy. In particular, for providers who already support DNSSEC, SMTP MTA-STS would additionally require they host a HTTPS webserver and obtain a CA-signed X.509 certificate for the recipient domain.
- o Security: DANE offers an advantage against policy-lookup DoS attacks; that is, while a DNSSEC-signed NXDOMAIN response to a DANE lookup authoritatively indicates the lack of a DANE record, such an option to authenticate policy non-existence does not exist when looking up a policy over plain DNS.

3. Policy Semantics

SMTP MTA-STS policies are distributed via a "well known" HTTPS endpoint in the Policy Domain. A corresponding TXT record in the DNS alerts sending MTAs to the presence of a policy file.

(Future implementations may move to alternate methods of policy discovery or distribution. See the section `_Future_ _Work_` for more discussion.)

The MTA-STS TXT record MUST specify the following fields:

- o "v": (plain-text, required). Currently only "STSv1" is supported.
- o "id": (plain-text, required). A short string used to track policy updates. This string MUST uniquely identify a given instance of a policy, such that senders can determine when the policy has been updated by comparing to the "id" of a previously seen policy, and must also match the "policy_id" value of the distributed policy.

A lenient parser SHOULD accept a policy file implementing a superset of this specification, in which case unknown values SHALL be ignored.

Policies MUST specify the following fields in JSON [[RFC4627](#)]

format:

- o "version": (plain-text, required). Currently only "STSv1" is supported.
- o "mode": (plain-text, required). If "enforce", the receiving MTA requests that messages be delivered only if they conform to the

STS policy. If "report" the receiving MTA requests that failure reports be delivered, as specified by the "rua" parameter.

- o "mx": MX patterns (list of plain-text MX match patterns, required). One or more comma-separated patterns matching the expected MX for this domain. For example, "["*.example.com", "*.example.net"]" indicates that mail for this domain might be handled by any MX whose hostname is a subdomain of "example.com" or "example.net". The semantics for these patterns should be the ones found in the "Checking of Wildcard Certificates" rules in [Section 6.4.3 of \[RFC6125\]](#).
- o "max_age": Max lifetime of the policy (plain-text integer seconds). Well-behaved clients SHOULD cache a policy for up to this value from last policy fetch time.
- o "policy_id": 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 "policy_id" of a previously seen policy.

A lenient parser SHOULD accept a policy file which is valid JSON implementing a superset of this specification, in which case unknown values SHALL be ignored.

[3.1. Formal Definition](#)

[3.1.1. TXT Record](#)

The formal definition of the "_mta_sts" TXT record, defined using [\[RFC5234\]](#), is as follows:

```
sts-text-record = sts-version *WSP %x3B *WSP sts-id

sts-version      = "v" *WSP "=" *WSP %x53 %x54          ; "STSV1"
                  %x53 %x76 %x31

sts-id           = "id" *WSP "=" *WSP 1*32(ALPHA / DIGIT)
```

[3.1.2. SMTP MTA-STS Policy](#)

The formal definition of the SMTP MTA-STS policy, using [\[RFC5234\]](#), is as follows:

```

sts-record      = WSP %x7B WSP ; { left curly bracket
                    sts-element ; comma-separated
                    [           ; list
                    WSP %x2c WSP ; of
                    sts-element ; sts-elements
                    ]
                    WSP %x7d WSP ; } right curly bracket

sts-element     = sts-version / sts-mode / sts-id / sts-mx / sts-max_age

sts-version     = %x22 "version" %x22 *WSP %x3a *WSP ; "version":
                    %x22 %x53 %x54 %x53 %x76 %x31 ; "STSV1"

sts-mode        = %x22 "mode" %x22 *WSP %x3a *WSP ; "mode":
                    %x22 ("report" / "enforce") %x22 ; "report"/"enforce"

sts-id          = %x22 "policy_id" %x22 *WSP %x3a *WSP ; "policy_id":
                    %x22 1*32(ALPHA / DIGIT) %x22 ; some chars

sts-mx          = %x22 "mx" $x22 *WSP %x3a *WSP ; "mx":
                    %x5B ; [
                    domain-match ; comma-separated list
                    [WSP %x2c domain-match WSP] ; of domain-matches
                    %x5B ; ]

sts-max_age     = %x22 "max_age" %x22 *WSP $x3a *WSP ; "max_age":
                    1*10DIGIT ; some digits

domain-match    = %x22 1*(dtext / "\"") *("." ; wildcard or label
                    1*dtext) %x22 ; with 0+ more labels

dtext           = ALPHA / DIGIT / %2D ; A-Z, a-z, 0-9, "-"

```

A size limitation in a sts-uri, if provided, is interpreted as a count of units followed by an OPTIONAL unit size ("k" for kilobytes, "m" for megabytes, "g" for gigabytes, "t" for terabytes). Without a unit, the number is presumed to be a basic byte count. Note that the units are considered to be powers of two; a kilobyte is 2^{10} , a megabyte is 2^{20} , etc.

[3.2. Policy Expirations](#)

In order to resist attackers inserting a fraudulent policy, SMTP MTA-STS policies are designed to be long-lived, with an expiry typically greater than two weeks. Policy validity is controlled by two separate expiration times: the lifetime indicated in the policy ("max_age=") and the TTL on the DNS record itself. The policy expiration will ordinarily be longer than that of the DNS TTL, and

senders SHOULD cache a policy (and apply it to all mail to the recipient domain) until the policy expiration.

An important consideration for domains publishing a policy is that senders will see a policy expiration as relative to the fetch of a policy cached by their recursive resolver. Consequently, a sender MAY treat a policy as valid for up to {expiration time} + {DNS TTL}. Publishers SHOULD thus continue to expect senders to apply old policies for up to this duration.

3.2.1. Policy Updates

Updating the policy requires that the owner make changes in two places: the "_mta_sts" RR record in the Policy Domain's DNS zone and at the corresponding HTTPS endpoint. In the case where the HTTPS endpoint has been updated but the TXT record has not been, senders will not know there is a new policy released and may thus continue to use old, previously cached versions. 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.

3.3. Policy Discovery & Authentication

Senders discover a recipient domain's STS policy, by making an attempt to fetch TXT records from the recipient domain's DNS zone with the name "_mta_sts". A valid TXT record presence in "_mta_sts.example.com" indicates that the recipient domain supports STS. To allow recipient domains to safely serve new policies, it is important that senders are able to authenticate a new policy retrieved for a recipient domain.

Web PKI is the mechanism used for policy authentication. In this mechanism, the sender fetches a HTTPS resource (policy) from a host at "policy.mta-sts" in the Policy Domain. The policy is served from a "well known" URI: "https://policy.mta-sts.example.com/.well-known/mta-sts/current". To consider the policy as valid, the "policy_id" field in the policy MUST match the "id" field in the DNS TXT record under "_mta_sts".

When fetching a new policy or updating a policy, the new policy MUST be fully authenticated (HTTPS certificate validation + peer verification) before use. A policy which has not ever been successfully authenticated MUST NOT be used to reject mail.

3.4. Policy Validation

When sending to an MX at a domain for which the sender has a valid and non-expired SMTP MTA-STS policy, a sending MTA honoring SMTP MTA-STS MUST validate that the recipient MX supports STARTTLS, and offers a valid PKIX based TLS certificate. The certificate presented by the receiving MX MUST be valid for the MX name and chain to a root CA that is trusted by the sending MTA. The certificate MUST have a CN or SAN matching the MX hostname (as described in [[RFC6125](#)]) and be non-expired.

3.5. Policy Application

When sending to an MX at a domain for which the sender has a valid non-expired SMTP MTA-STS policy, a sending MTA honoring SMTP MTA-STS MAY apply the result of a policy validation one of two ways:

- o "report": In this mode, sending MTAs merely send a report to the designated report address indicating policy application failures. This can be done "offline", i.e. based on the MTA logs, and is thus a suitable low-risk option for MTAs who wish to enhance transparency of TLS tampering without making complicated changes to production mail-handling infrastructure.
- o "enforce": In this mode, sending MTAs SHOULD treat STS policy failures, in which the policy action is "reject", as a mail delivery error, and SHOULD terminate the SMTP connection, not delivering any more mail to the recipient MTA.

In "enforce" mode, however, sending MTAs MUST first check for a new authenticated policy before actually treating a message failure as fatal.

Thus the control flow for a sending MTA that does online policy application consists of the following steps:

1. Check for cached non-expired policy. If none exists, fetch the latest, authenticate and cache it.
2. Validate recipient MTA against policy. If valid, deliver mail.
3. If not valid and the policy specifies reporting, generate report.
4. If not valid and policy specifies rejection, perform the following steps:
 - * Check for a new (non-cached) authenticated policy.

- * If one exists and the new policy is different, update the current policy and go to step 2.
- * If one exists and the new policy is same as the cached policy, treat the delivery as a failure.
- * If none exists and cached policy is not expired, treat the delivery as a failure.

Understanding the details of step 4 is critical to understanding the behavior of the system as a whole.

Remember that each policy has an expiration time (which SHOULD be long, on the order of days or months) and a validation method. With these two mechanisms and the procedure specified in step 4, recipients who publish a policy have, in effect, a means of updating a cached policy at arbitrary intervals, without the risks (of a man-in-the-middle attack) they would incur if they were to shorten the policy expiration time.

4. Failure Reporting

Aggregate statistics on policy failures MAY be reported using the "TLSRPT" reporting specification (TODO: Add Ref).

5. IANA Considerations

There are no IANA considerations at this time.

6. Security Considerations

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

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

SMTP Strict Transport Security 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 out of scope of this threat model.

Since we use DNS TXT record for policy discovery, an attacker who is able to block DNS responses can suppress the discovery of an STS Policy, making the Policy Domain appear not to have an STS Policy. The caching model described in `_Policy_` `_Expirations_` is designed to resist this attack, and there is discussion in the `_Future_` `_Work_` section around future distribution mechanisms that are robust against this attack.

7. Future Work

The authors would like to suggest multiple considerations for future discussion.

- o Certificate pinning: One potential improvement in the robustness of the certificate validation methods discussed would be the deployment of public-key pinning as defined for HTTP in [[RFC7469](#)]. A policy extension supporting these semantics would enable Policy Domains to specify certificates that **MUST** appear in the MX certificate chain, thus providing resistance against compromised CA or DNSSEC zone keys.
- o Policy distribution: As with Certificate Transparency ([[RFC6962](#)]), it may be possible to provide a verifiable log of policy `_observations_` (meaning which policies have been observed for a given Policy Domain). This would provide insight into policy spoofing or faked policy non-existence. This may be particularly useful for Policy Domains not using DNSSEC, since it would provide sending MTAs an authoritative source for whether a policy is expected for a given domain.
- o Receive-from restrictions: Policy publishers may wish to also indicate to domains `_receiving_` mail from the Policy Domain that all such mail is expected to be sent via TLS. This may allow policy publishers to receive reports indicating sending MTA misconfigurations. However, the security properties of a "receiver-enforced" system differ from those of the current design; in particular, an active man-in-the-middle attacker may be able to exploit misconfigured sending MTAs in a way that would not be possible today with a sender-enforced model.

- o Cipher and TLS version restrictions: Policy publishers may also wish to restrict TLS negotiation to specific ciphers or TLS versions.

8. Appendix 1: Validation Pseudocode

```
policy = policy_from_cache()
if not policy or is_expired(policy):
    policy = policy_from_https_endpoint() // fetch and authenticate!
    update_cache = true
if policy:
    if invalid_mx_or_tls(policy): // check MX and TLS cert
        if rua:
            generate_report()
        if p_reject():
            policy = policy_from_https_endpoint() // fetch and authenticate #2!
            update_cache = true
            if invalid_mx_or_tls(policy):
                reject_message()
                update_cache = false
if update_cache:
    cache(policy)
```

9. Appendix 2: Domain Owner STS example record

9.1. Example 1

The owner of example.com wishes to begin using STS with a policy that will solicit aggregate feedback from receivers without affecting how the messages are processed, in order to:

- o Verify the identity of MXs that handle mail for this domain
- o Confirm that its legitimate messages are sent over TLS
- o Verify the validity of the certificates
- o Determine how many messages would be affected by a strict policy

DNS STS policy indicator TXT record:

```
_mta_sts IN TXT ( "v=STSV1; id=randomstr;" )
```

STS policy served from HTTPS endpoint of the policy (recipient) domain, and is authenticated using Web PKI mechanism. The policy is fetched using HTTP GET method.


```
{  
  "version": "STSv1",  
  "mode": "report",  
  "policy_id": "randomstr",  
  "mx": ["*.mail.example.com"],  
  "max_age": 123456  
}
```

The policy is authenticated using Web PKI mechanism.

10. Appendix 3: DEEP Registration Elements

Name: mx-mismatch

Description: This indicates that the MX resolved for the recipient domain did not match the MX constraint specified in the policy.

Intended Usage: COMMON

Reference: RFC XXXX (this document once published)

Submitter: Authors of this document

Change Controller: IESG

Name: certificate-name-mismatch

Description: This indicates that the subject CNAME/SAN in the certificate presented by the receiving MX did not match the MX hostname

Intended Usage: COMMON

Reference: RFC XXXX (this document once published)

Submitter: Authors of this document

Change Controller: IESG

Name: invalid-certificate

Description: This indicates that the certificate presented by the receiving MX did not validate according to the policy validation constraint. (Either it was not signed by a trusted CA or did not match the DANE TLSA record for the recipient MX.)

Intended Usage: COMMON

Reference: RFC XXXX (this document once published)

Submitter: Authors of this document

Change Controller: IESG

Name: certificate-name-constraints-not-permitted

Description: The certificate request contains a name that is not listed as permitted in the name constraints extension of the cert issuer.

Intended Usage: COMMON

Reference: RFC XXXX (this document once published)

Submitter: Authors of this document

Change Controller: IESG

Name: certificate-name-constraints-excluded

Description: The certificate request contains a name that is listed as

excluded in the name constraints extension of the issuer.

Intended Usage: COMMON

Reference: RFC XXXX (this document once published)

Submitter: Authors of this document

Change Controller: IESG

Name: expired-certificate

Description: This indicates that the certificate has expired.

Intended Usage: COMMON

Reference: RFC XXXX (this document once published)

Submitter: Authors of this document

Change Controller: IESG

Name: starttls-not-supported

Description: This indicates that the recipient MX did not support STARTTLS.

Intended Usage: COMMON

Reference: RFC XXXX (this document once published)

Submitter: Authors of this document

Change Controller: IESG

Name: tlsa-invalid

Description: This indicates a validation error for Policy Domain specifying "tlsa" validation.

Intended Usage: COMMON

Reference: RFC XXXX (this document once published)

Submitter: Authors of this document

Change Controller: IESG

Name: dnssec-invalid

Description: This indicates a failure to validate DNS records for a Policy Domain specifying "tlsa" validation or "dnssec" authentication.

Intended Usage: COMMON

Reference: RFC XXXX (this document once published)

Submitter: Authors of this document

Change Controller: IESG

Name: sender-does-not-support-validation-method

Description: This indicates the sending system can never validate using the requested validation mechanism.

Intended Usage: COMMON

Reference: RFC XXXX (this document once published)

Submitter: Authors of this document

Change Controller: IESG

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>>.
- [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>>.
- [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>>.
- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", [RFC 4627](#), DOI 10.17487/RFC4627, July 2006, <<http://www.rfc-editor.org/info/rfc4627>>.
- [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>>.
- [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>>.
- [RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", [RFC 6962](#), DOI 10.17487/RFC6962, June 2013, <<http://www.rfc-editor.org/info/rfc6962>>.
- [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>>.
- [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>>.

Authors' Addresses

Daniel Margolis
Google, Inc

Email: dmargolis (at) google.com

Mark Risher
Google, Inc

Email: riser (at) google (dot com)

Nicolas Lidzborski
Google, Inc

Email: nlidz (at) google (dot com)

Wei Chuang
Google, Inc

Email: weihaw (at) google (dot com)

Brandon Long
Google, Inc

Email: blong (at) google (dot com)

Binu Ramakrishnan
Yahoo!, Inc

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

Alexander Brotman
Comcast, Inc

Email: alexander_brotman (at) cable.comcast (dot com)

Janet Jones
Microsoft, Inc

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

Franck Martin
LinkedIn

Email: `fmartin (at) linkedin (dot) com`

Klaus Umbach
l&l Mail & Media Development & Technology GmbH

Email: `klaus.umbach (at) lund1 (dot) de`

Markus Laber
l&l Mail & Media Development & Technology GmbH

Email: `markus.laber (at) lund1 (dot) de`