Using TLS in Applications
Internet-Draft
Intended status: Standards Track
Expires: September 19, 2016

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
March 18, 2016

SMTP Strict Transport Security
draft-margolis-smtp-sts-00

Abstract

SMTP 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 sending SMTP servers to 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 September 19, 2016.

Table of Contents

1. Introduction 2
1.1. Terminology 3
2. Related Technologies 4
2.1. Differences from DANE 4
2.2. Advantages When Used with DANE 4
2.3. Advantages When Used Without DANE 4
2.4. Disadvantages When Used Without DANE 5
3. Policy Semantics 5
3.1. Formal Definition 6
3.2. Policy Expirations 8
3.3. Policy Authentication 8
3.4. Policy Validation 9
3.5. Policy Application 9
4. Failure Reporting 10
5. IANA Considerations 12
6. Security Considerations 12
7. Future Work 13
8. Appendix 1: Validation Pseudocode 14
9. Appendix 2: Domain Owner STS example record 14
10. Appendix 3: XML Schema for Failure Reports 14
11. Appendix 4: Example report 16
12. Normative References 17
Authors' Addresses 18

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:

- whether MTAs sending mail to this domain can expect TLS support
- how MTAs can validate the TLS server certificate presented during mail delivery
- what an implementing sender should do with messages when TLS cannot be 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 authentication: how to determine the authenticity of a published policy delivered via DNS
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:

- 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.

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

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, and specifies a method for reporting TLS negotiation failures.

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 and a trust-on-first-use (TOFU) approach to avoid interception. The TOFU model allows a degree of security similar to that of HPKP [RFC7469], reducing the complexity but without the guarantees on first use offered by DNSSEC. (For a thorough discussion of this trade-off, see the section _Security_ _Considerations_.)

In addition, SMTP STS introduces a mechanism for failure reporting and a report-only mode, enabling progressive roll-out and auditing for compliance.

2.2. Advantages When Used with DANE

SMTP STS can be deployed for a recipient domain that also publishes a DANE TLSA record for SMTP. In these cases, the SMTP STS policy can additionally declare a process for failure reporting.

2.3. Advantages When Used Without DANE

When deployed without a DANE TLSA record, SMTP STS offers the following advantages compared to DANE:

- _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 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.

- _Incrementalism:_ DANE does not provide a reporting mechanism and does not have a concept of "report-only" for failures; as a

result, a service provider has no choice but to "flip the switch" and affect the entire mail stream at once.

2.4. Disadvantages When Used Without DANE

When deployed alone (i.e. without a DANE record, and using Web PKI for certificate verification), SMTP STS offers the following disadvantages compared to DANE:

- Infrastructure: DANE may be easier for some providers to deploy. In particular, for providers who already support DNSSEC, SMTP STS would additionally require they obtain a CA-signed x509 certificate for the recipient domain.

- Security: DANE offers an advantage against policy-lookup DoS attacks; that is, while a DNSSEC-signed NX 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 STS policies are distributed at the Policy Domain either through a new resource record, or as TXT records (similar to DMARC policies) under the name "_smtp_sts." (Current implementations deploy as TXT records.) For example, for the Policy Domain "example.com", the recipient's SMTP STS policy can be retrieved from "_smtp_sts.example.com."

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

Policies MUST specify the following fields:

- v: Version (plain-text, required). Currently only "STS1" is supported.

- to: TLS-Only (plain-text, required). If "true," the receiving MTA requests that messages be delivered only if they conform to the STS policy. If "false," the receiving MTA requests that failure reports be delivered, as specified by the "rua" parameter.

- mx: MX patterns (comma-separated 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."

- a: The mechanism to use to authenticate this policy itself. See the section _Policy_ _Authentication_ for more details. Possible values are:
  - webpki:URI, where URI points to an HTTPS resource at the recipient domain that serves the same policy text.
  - dnssec: Indicating that the policy is expected to be served over DNSSEC.
- c: Constraints on the recipient MX's TLS certificate (plain-text, required). See the section _Policy_ _Validation_ for more details. Possible values are:
  - webpki: Indicating that the TLS certificate presented by the recipient MX will be validated according to the "web PKI" mechanism.
  - tlsa: Indicating that the TLS certificate presented by the recipient MX will match a (presumed to exist) DANE TLSA record.
- e: 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.
- rua: Address to which aggregate feedback MAY be sent (comma-separated plain-text list of email addresses, optional). For example, "mailto:postmaster@example.com" from [RFC3986].

3.1. Formal Definition

The formal definition of the SMTP STS format, using [RFC5234], is as follows:

```
   sts-uri         = URI [ "!" 1*DIGIT [ "k" / "m" / "g" / "t" ] ]
                     ; "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

   sts-record      = sts-version sts-sep sts-to
                     [sts-sep sts-mx]
                     [sts-sep sts-a]
                     [sts-sep sts-c]
                     [sts-sep sts-e]
                     [sts-sep sts-auri]
                     [sts-sep]
                     ; components other than sts-version and
                     ; sts-to may appear in any order

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

   sts-sep         = *WSP %x3b *WSP

   sts-to          = "to" *WSP "=" *WSP ( "true" / "false" )

   sts-mx          = "mx" *WSP "=" *WSP sts-domain-list

   sts-domain-list = (domain-match *("," domain-match))

   domain-match    =  ["*."] 1*dtext *("." 1*dtext)

   dtext           =  %d30-39 /          ; 0-9
                      %d41-5A /          ; a-z
                      %61-7A /           ; A-Z
                      %2D                ; "-"

   sts-a           = "a" *WSP "=" *WSP ( URI / "dnssec")

   sts-c           = "c" *WSP "=" *WSP ( "webpki" / "tlsa")

   sts-e           = "e" *WSP "=" *WSP 1*6DIGIT

   sts-auri        = "rua" *WSP "=" *WSP
                      sts-uri *(*WSP "," *WSP sts-uri)
```

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 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 ("e=") 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.

3.3. Policy Authentication

The security of a domain implementing an SMTP STS policy against an active man-in-the-middle depends primarily upon the long-lived caching of policies. However, to allow recipient domains to safely serve new policies _prior_ to the expiration of a cached policy, and to prevent long-term (either malicious or active) denials of service, it is important that senders are able to validate a new policy retrieved for a recipient domain. There are two supported mechanisms for policy validation:

- Web PKI: In this mechanism, indicated by the "webpki" value of the "a" field, the sender fetches a HTTPS resource from the URI indicated. For example, a=webpki:<https://example.com/.well-known/smtp-sts/current> indicates that the sender should fetch the resource <https://example.com/.well-known/smtp-sts/current>. In order for the policy to be valid, the HTTP response body served at this resource MUST exactly match the policy initially loaded via the DNS TXT method, and MUST be served from an HTTPS endpoint at the domain matching that of the recipient domain. (As this RFC progress, the authors intend to register .well-known/smtp-sts. See [RFC5785]. See _Future_ _Work_ for more information.)

- DNSSEC: In this mechanism, indicated by the "dnssec" value of the "a" field, the sender MUST retrieve the policy via a DNSSEC signed response for the _smtp_sts TXT record.

When fetching a new policy when one is not already known, or when fetching a policy for a domain with an expired policy, unauthenticated policies MUST be trusted and honored. When fetching

a policy and authenticating it, as described in detail in _Policy_ _Application_, policies will be authenticated using the mechanism specified by the existing cached policy.

Note, however, as described in detail in _Policy_ _Application_, that new policies MUST NOT be considered as valid if they do not validate on first application. That is, a freshly fetched (and unused) policy that has not successfully been applied MUST be disregarded.

3.4. Policy Validation

When sending to an MX at a domain for which the sender has a valid and non-expired SMTP STS policy, a sending MTA honoring SMTP STS SHOULD validate that the recipient MX supports STARTTLS and offers a TLS certificate which is valid according to the semantics of the SMTP STS policy. Policies can specify certificate validity in one of two ways by setting the value of the "c" field in the policy description.

- Web PKI: When the "c" field is set to "webpki", 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.

- DANE TLSA: When the "c" field is set to "tlsa", the receiving MX MUST be covered by a DANE TLSA record for the recipient domain, and the presented certificate MUST be valid according to that record (as described by [RFC7672]).

A sending MTA who does not support the validation method required--for example, an MTA that does not have a DNSSEC-compatible resolver--MUST behave as though the policy did not validate. As described in the section on _Policy_ _Application_, a policy which has not ever been successfully validated MUST not be used to reject mail.

3.5. Policy Application

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

- Report-only: 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.

- Enforced: 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 enforced 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 and cache it.

2. Validate recipient MTA against policy. If valid, deliver mail.

3. If policy invalid and policy specifies reporting, generate report.

4. If policy invalid and policy specifies rejection, perform the following steps:

    * Check for a new (non-cached) _authenticated_ policy. If one exists, update the current policy and go to step 1.

    * If none exists or the newly fetched policy also fails, 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 to the URI indicated in the "rua" field of the policy. SMTP STS reports contain information about policy failures to allow diagnosis of misconfigurations and malicious activity.

(There may also be a need for enabling more detailed "forensic" reporting during initial stages of a deployment. To address this, the authors consider the possibility of an optional additional "forensic reporting mode" in which more details--such as certificate chains and MTA banners--may be reported. See the section _Future_ _Work_ for more details.)

Aggregate reports contain the following fields:

- The SMTP STS policy applied (as a string)
- The beginning and end of the reporting period

Repeated records contain the following fields:

- Failure type: This list will start with the minimal set below, and is expected to grow over time based on real-world experience. The initial set is:
  - mx-mismatch: This indicates that the MX resolved for the recipient domain did not match the MX constraint specified in the policy.
  - certificate-mismatch: This indicates that the certificate presented by the receiving MX did not match the MX hostname
  - invalid-certificate: 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.)
  - expired-certificate: This indicates that the certificate has expired.
  - starttls-not-supported: This indicates that the recipient MX did not support STARTTLS.
  - tlsa-invalid: This indicates a validation error for Policy Domain specifying "tlsa" validation.
  - dnssec-invalid: This indicates a failure to validate DNS records for a Policy Domain specifying "tlsa" validation or "dnssec" authentication.
  - sender-does-not-support-validation-method: This indicates the sending system can never validate using the requested validation mechanism.

- Count: The number of times the error was encountered.

- Hostname: The hostname of the recipient MX.

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

When sending failure reports, sending MTAs MUST NOT honor SMTP STS or DANE TLSA failures.

5. IANA Considerations

The ".well-known" URI for Policy Domains to host their STS Policies will be registered by following the procedure documented in [RFC5785] (i.e. sending a request to the "wellknown-uri-review@ietf.org" mailing list for review and comment). The proposed URI-suffix is "smtp-sts".

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:

- 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 to the legitimate recipient server (for example, by altering BGP routing tables).

SMTP Strict Transport Security relies on certificate validation via either TLS identity checking [RFC6125] or DANE TLSA [RFC7672]. 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.

In the WebPKI constraint mode, an attacker who is able to block DNS responses can suppress the delivery 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.

- 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 resistence against compromised CA or DNSSEC zone keys.

- 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.

- 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.

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

In addition, the authors leave currently open the following details:

- Whether and how more detailed "forensic reporting" should be accomplished, as discussed in the section _Failure_ _Reporting_.

- The registration of the .well-known/smtp-sts URI as per [RFC5785].

8. Appendix 1: Validation Pseudocode

```
  policy = policy_from_cache()
  if not policy or is_expired(policy):
    policy = policy_from_dns()  // 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_dns()  // 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

The owner 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:

* Confirm that its legitimate messages are sent over TLS

* Verify the validity of the certificates

* Verify what cyphers are in use

* Determine how many messages would be affected by a strict policy

```
_smtp_sts  IN TXT ( "v=STS1; to=false; "
                    "rua=mailto:sts-feedback@example.com " )
```

10. Appendix 3: XML Schema for Failure Reports

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.example.org/smtp-sts-xml/0.1"
    xmlns:tns="http://www.example.org/smtp-sts-xml/0.1">
   <!-- The time range in UTC covered by messages in this report,
        specified in seconds since epoch. -->
   <xs:complexType name="DateRangeType">
     <xs:all>
       <xs:element name="begin" type="xs:integer"/>
```

```
        <xs:element name="end" type="xs:integer"/>
      </xs:all>
    </xs:complexType>

    <!-- Report generator metadata. -->
    <xs:complexType name="ReportMetadataType">
      <xs:sequence>
        <xs:element name="org_name" type="xs:string"/>
        <xs:element name="email" type="xs:string"/>
        <xs:element name="extra_contact_info" type="xs:string"
                    minOccurs="0"/>
        <xs:element name="report_id" type="xs:string"/>
        <xs:element name="date_range" type="tns:DateRangeType"/>
      </xs:sequence>
    </xs:complexType>


    <!-- The constraints applied in a policy -->
    <xs:simpleType name="ConstraintType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="WebPKI"/>
        <xs:enumeration value="TLSA"/>
      </xs:restriction>
    </xs:simpleType>

    <!-- The policy that was applied at send time. -->
    <xs:complexType name="AppliedPolicyType">
      <xs:all>
        <xs:element name="domain" type="xs:string"/>
        <xs:element name="mx" type="xs:string"
            minOccurs="1" />
        <xs:element name="constraint" type="tns:ConstraintType"/>
      </xs:all>
    </xs:complexType>

    <!-- The possible failure types applied in a policy -->
    <xs:simpleType name="FailureType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="MxMismatch"/>
        <xs:enumeration value="InvalidCertificate"/>
        <xs:enumeration value="ExpiredCertificate"/>
        <xs:enumeration value="StarttlsNotSupported"/>
        <xs:enumeration value="TlsaInvalid"/>
        <xs:enumeration value="DnssecInvalid"/>
        <xs:enumeration value="SenderDoesNotSupportValidationMethod"/>
      </xs:restriction>
    </xs:simpleType>
```

```
    <!-- The possible enforcement level: whether the reporter also drops
         messages -->
    <xs:simpleType name="EnforcementLevelType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="ReportOnly"/>
        <xs:enumeration value="Reject"/>
      </xs:restriction>
    </xs:simpleType>

    <!-- Record for individual failure types. -->
    <xs:complexType name="FailureRecordType">
      <xs:all>
        <xs:element name="failure" type="tns:FailureType"/>
        <xs:element name="count" type="xs:integer"/>
        <xs:element name="hostname" type="xs:string"/>
        <xs:element name="connectedIp" type="xs:string" minOccurs="0"/>
        <xs:element name="sourceIp" type="xs:string" minOccurs="0"/>
      </xs:all>
    </xs:complexType>

     <!-- Parent -->
    <xs:element name="feedback">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="version"
                      type="xs:decimal"/>
          <xs:element name="report_metadata"
                      type="tns:ReportMetadataType"/>
          <xs:element name="applied_policy"
                      type="tns:AppliedPolicyType"/>
    <xs:element name="enforcement_level"
    type="tns:EnforcementLevelType"/>
          <xs:element name="record" type="tns:FailureRecordType"
                      maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
 </xs:schema>
```

11. Appendix 4: Example report

```
<feedback xmlns="http://www.example.org/smtp-sts-xml/0.1">
  <version>1</version>
  <report_metadata>
    <org_name>Company XYZ</org_name>
    <email>sts-reporting@company.com</email>
    <extra_contact_info></extra_contact_info>
    <report_id>12345</report_id>
    <date_range><begin>1439227624</begin>
    <end>1439313998</end></date_range>
    </report_metadata>
  <applied_policy>
    <domain>company.com</domain>
    <mx>*.mx.mail.company.com</mx>
    <constraint>WebPKI</constraint>
  </applied_policy>
   <enforcement_level>ReportOnly</enforcement_level>
  <record>
      <failure>ExpiredCertificate</failure>
      <count>13128</count>
      <hostname>mta7.am0.yahoodns.net.</hostname>
      <connectedIp> 98.136.216.25</connectedIp>
  </record>
  <record>
      <failure>StarttlsNotSupported</failure>
      <count>19</count>
      <hostname>mta7.am0.yahoodns.net.</hostname>
      <connectedIp>98.22.33.99</connectedIp>
  </record>
</feedback>
```

12. 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>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[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>.

[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>.

[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: risher (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
1&1 Mail & Media Development & Technology GmbH

Email: klaus.umbach (at) 1und1 (dot de)

Markus Laber
1&1 Mail & Media Development & Technology GmbH

Email: markus.laber (at) 1und1 (dot de)