SMTP Service Extension for Authentication

Status of This Memo

Copyright Notice

Abstract

   This document defines a Simple Mail Transport Protocol (SMTP)
   extension whereby an SMTP client may indicate an authentication
   mechanism to the server, perform an authentication protocol exchange,
   and optionally negotiate a security layer for subsequent protocol
   interactions during this session. This extension includes a profile
   of the Simple Authentication and Security Layer (SASL) for SMTP.

   This document obsoletes RFC 2554.

Table of Contents

1. Introduction

   This document defines a Simple Mail Transport Protocol (SMTP)
   extension whereby an SMTP client may indicate an authentication
   mechanism to the server, perform an authentication protocol exchange,
   optionally negotiate a security layer for subsequent protocol
   interactions during this session and, during a mail transaction,
   optionally specify a mailbox associated with the identity that
   submitted the message to the mail delivery system.

   This extension includes a profile of the Simple Authentication and
   Security Layer (SASL) for SMTP.

   When compared to RFC 2554, this document deprecates use of the 538
   response code, adds a new Enhanced Status Code, adds a requirement to
   support SASLprep profile for preparing authorization identities,
   recommends use of RFC 3848 transmission types in the Received trace
   header field, and clarifies interaction with SMTP PIPELINING
   [PIPELINING] extension.

2. How to Read 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 [KEYWORDS].

   In examples, "C:" and "S:" indicate lines sent by the client and
   server, respectively.

3. The Authentication Service Extension

   1. The name of this [SMTP] service extension is "Authentication".

   2. The EHLO keyword value associated with this extension is "AUTH".

   3. The AUTH EHLO keyword contains as a parameter a space-separated
       list of the names of available [SASL] mechanisms. The list of
       available mechanisms MAY change after a successful STARTTLS
       command [SMTP-TLS].

   4. A new [SMTP] verb "AUTH" is defined.

   5. An optional parameter using the keyword "AUTH" is added to the
       MAIL FROM command, and extends the maximum line length of the
       MAIL FROM command by 500 characters.

   6. This extension is appropriate for the submission protocol
       [SUBMIT].

4. The AUTH Command

   AUTH mechanism [initial-response]

      Arguments:
         mechanism: A string identifying a [SASL] authentication
         mechanism.

         initial-response: An optional initial client response. If
         present, this response MUST be encoded as described in Section
         4 of [BASE64] or contain a single character "=".

      Restrictions:
         After an AUTH command has been successfully completed, no more
         AUTH commands may be issued in the same session. After a
         successful AUTH command completes, a server MUST reject any
         further AUTH commands with a 503 reply.

         The AUTH command is not permitted during a mail transaction.
         An AUTH command issued during a mail transaction MUST be
         rejected with a 503 reply.

      Discussion:
         The AUTH command initiates a [SASL] authentication exchange
         between the client and the server. The client identifies the
         SASL mechanism to use with the first parameter of the AUTH
         command. If the server supports the requested authentication
         mechanism, it performs the SASL exchange to authenticate the

user. Optionally, it also negotiates a security layer for
subsequent protocol interactions during this session. If the
requested authentication mechanism is invalid (e.g., is not
supported or requires an encryption layer), the server rejects
the AUTH command with a 504 reply. If the server supports the
[ESMTP-CODES] extension, it SHOULD return a 5.5.4 enhanced
response code.

The SASL authentication exchange consists of a series of
server challenges and client responses that are specific to
the chosen [SASL] mechanism.

A server challenge is sent as a 334 reply with the text part
containing the [BASE64] encoded string supplied by the SASL
mechanism. This challenge MUST NOT contain any text other
than the BASE64 encoded challenge.

A client response consists of a line containing a [BASE64]
encoded string. If the client wishes to cancel the
authentication exchange, it issues a line with a single "*".
If the server receives such a response, it MUST reject the
AUTH command by sending a 501 reply.

The optional initial response argument to the AUTH command is
used to save a round-trip when using authentication mechanisms
that support an initial client response. If the initial
response argument is omitted and the chosen mechanism requires
an initial client response, the server MUST proceed as defined
in Section 5.1 of [SASL]. In SMTP, a server challenge that
contains no data is defined as a 334 reply with no text part.
Note that there is still a space following the reply code, so
the complete response line is "334 ".

Note that the AUTH command is still subject to the line length
limitations defined in [SMTP]. If use of the initial response
argument would cause the AUTH command to exceed this length,
the client MUST NOT use the initial response parameter (and
instead proceed as defined in Section 5.1 of [SASL]).

If the client is transmitting an initial response of zero
length, it MUST instead transmit the response as a single
equals sign ("="). This indicates that the response is
present, but contains no data.

If the client uses an initial-response argument to the AUTH
command with a SASL mechanism in which the client does not
begin the authentication exchange, the server MUST reject the

AUTH command with a 501 reply. Servers using the enhanced
status codes extension [ESMTP-CODES] SHOULD return an enhanced
status code of 5.7.0 in this case.

If the server cannot [BASE64] decode any client response, it
MUST reject the AUTH command with a 501 reply (and an enhanced
status code of 5.5.2). If the client cannot BASE64 decode any
of the server's challenges, it MUST cancel the authentication
using the "*" response. In particular, servers and clients
MUST reject (and not ignore) any character not explicitly
allowed by the BASE64 alphabet, and MUST reject any sequence
of BASE64 characters that contains the pad character ('=')
anywhere other than the end of the string (e.g., "=AAA" and
"AAA=BBB" are not allowed).

Note that these [BASE64] strings can be much longer than
normal SMTP commands. Clients and servers MUST be able to
handle the maximum encoded size of challenges and responses
generated by their supported authentication mechanisms. This
requirement is independent of any line length limitations the
client or server may have in other parts of its protocol
implementation. (At the time of writing of this document,
12288 octets is considered to be a sufficient line length
limit for handling of deployed authentication mechanisms.)
If, during an authentication exchange, the server receives a
line that is longer than the server's authentication buffer,
the server fails the AUTH command with the 500 reply. Servers
using the enhanced status codes extension [ESMTP-CODES] SHOULD
return an enhanced status code of 5.5.6 in this case.

The authorization identity generated by this [SASL] exchange
is a "simple username" (in the sense defined in [SASLprep]),
and both client and server SHOULD (*) use the [SASLprep]
profile of the [StringPrep] algorithm to prepare these names
for transmission or comparison. If preparation of the
authorization identity fails or results in an empty string
(unless it was transmitted as the empty string), the server
MUST fail the authentication.

(*) Note: Future revision of this specification may change this
requirement to MUST. Currently, the SHOULD is used in order to
avoid breaking the majority of existing implementations.

If the server is unable to authenticate the client, it SHOULD reject
the AUTH command with a 535 reply unless a more specific error code
is appropriate. Should the client successfully complete the
exchange, the SMTP server issues a 235 reply. (Note that the SMTP
protocol doesn't support the SASL feature of returning additional

data with a successful outcome.) These status codes, along with
others defined by this extension, are discussed in Section 6 of this
document.

If a security layer is negotiated during the SASL exchange, it takes
effect for the client on the octet immediately following the CRLF
that concludes the last response generated by the client. For the
server, it takes effect immediately following the CRLF of its success
reply.

When a security layer takes effect, the SMTP protocol is reset to the
initial state (the state in SMTP after a server issues a 220 service
ready greeting). The server MUST discard any knowledge obtained from
the client, such as the EHLO argument, which was not obtained from
the SASL negotiation itself. Likewise, the client MUST discard any
knowledge obtained from the server, such as the list of SMTP service
extensions, which was not obtained from the SASL negotiation itself.
(Note that a client MAY compare the advertised SASL mechanisms before
and after authentication in order to detect an active down-
negotiation attack).

The client SHOULD send an EHLO command as the first command after a
successful SASL negotiation that results in the enabling of a
security layer.

When an entity (whether it is the client or the server end) is
sending data, and both [TLS] and SASL security layers are in effect,
the TLS encoding MUST be applied after the SASL encoding, regardless
of the order in which the layers were negotiated.

The service name specified by this protocol's profile of SASL is
"smtp". This service name is also to be used for the [SUBMIT]
protocol.

If an AUTH command fails, the client MAY proceed without
authentication. Alternatively, the client MAY try another
authentication mechanism or present different credentials by issuing
another AUTH

Note: A server implementation MUST implement a configuration in which
it does NOT permit any plaintext password mechanisms, unless either
the STARTTLS [SMTP-TLS] command has been negotiated or some other
mechanism that protects the session from password snooping has been
provided. Server sites SHOULD NOT use any configuration which
permits a plaintext password mechanism without such a protection
mechanism against password snooping.

To ensure interoperability, client and server implementations of this
extension MUST implement the [PLAIN] SASL mechanism running over TLS
[TLS] [SMTP-TLS]. See also Section 15 for additional requirements on
implementations of [PLAIN] over [TLS].

Note that many existing client and server implementations implement
CRAM-MD5 [CRAM-MD5] SASL mechanism. In order to ensure
interoperability with deployed software, new implementations MAY
implement it; however, implementations should be aware that this SASL
mechanism doesn't provide any server authentication. Note that at
the time of writing of this document the SASL Working Group is
working on several replacement SASL mechanisms that provide server
authentication and other features.

When the AUTH command is used together with the [PIPELINING]
extension, it MUST be the last command in a pipelined group of
commands. The only exception to this rule is when the AUTH command
contains an initial response for a SASL mechanism that allows the
client to send data first, the SASL mechanism is known to complete in
one round-trip, and a security layer is not negotiated by the client.
Two examples of such SASL mechanisms are PLAIN [PLAIN] and EXTERNAL
[SASL].

4.1. Examples

Here is an example of a client attempting AUTH using the [PLAIN] SASL
mechanism under a TLS layer, and making use of the initial client
response:

```
S: 220-smtp.example.com ESMTP Server
C: EHLO client.example.com
S: 250-smtp.example.com Hello client.example.com
S: 250-AUTH GSSAPI DIGEST-MD5
S: 250-ENHANCEDSTATUSCODES
S: 250 STARTTLS
C: STARTTLS
S: 220 Ready to start TLS
  ... TLS negotiation proceeds, further commands
      protected by TLS layer ...
C: EHLO client.example.com
S: 250-smtp.example.com Hello client.example.com
S: 250 AUTH GSSAPI DIGEST-MD5 PLAIN
C: AUTH PLAIN dGVzdAB0ZXN0ADEyMzQ=
S: 235 2.7.0 Authentication successful
```

Here is another client that is attempting AUTH PLAIN under a TLS
layer, this time without the initial response. Parts of the
negotiation before the TLS layer was established have been omitted:

```
       ... TLS negotiation proceeds, further commands
           protected by TLS layer ...
    C: EHLO client.example.com
    S: 250-smtp.example.com Hello client.example.com
    S: 250 AUTH GSSAPI DIGEST-MD5 PLAIN
    C: AUTH PLAIN
     (note: there is a single space following the 334
      on the following line)
    S: 334
    C: dGVzdAB0ZXN0ADEyMzQ=
    S: 235 2.7.0 Authentication successful
```

   Here is an example using CRAM-MD5 [CRAM-MD5], a mechanism in which
   the client does not begin the authentication exchange, and includes a
   server challenge:

```
    S: 220-smtp.example.com ESMTP Server
    C: EHLO client.example.com
    S: 250-smtp.example.com Hello client.example.com
    S: 250-AUTH DIGEST-MD5 CRAM-MD5
    S: 250-ENHANCEDSTATUSCODES
    S: 250 STARTTLS
    C: AUTH CRAM-MD5
    S: 334 PDQxOTI5NDIzNDEuMTI4Mjg0NzJAc291cmNlZm91ci5hbmRyZXcuY21lLmVk
       dT4=
    C: cmpzMyBlYzNhNTlmZWQzOTVhYmExZWM2MzY3YzRmNGI0MWFjMA==
    S: 235 2.7.0 Authentication successful
```

   Here is an example of a client attempting AUTH EXTERNAL under TLS,
   using the derived authorization ID (and thus a zero-length initial
   client response).

```
    S: 220-smtp.example.com ESMTP Server
    C: EHLO client.example.com
    S: 250-smtp.example.com Hello client.example.com
    S: 250-AUTH GSSAPI DIGEST-MD5
    S: 250-ENHANCEDSTATUSCODES
    S: 250 STARTTLS
    C: STARTTLS
    S: 220 Ready to start TLS
      ... TLS negotiation proceeds, further commands
          protected by TLS layer ...
    C: EHLO client.example.com
    S: 250-smtp.example.com Hello client.example.com
    S: 250 AUTH EXTERNAL GSSAPI DIGEST-MD5 PLAIN
    C: AUTH EXTERNAL =
    S: 235 2.7.0 Authentication successful
```

5. The AUTH Parameter to the MAIL FROM command

   AUTH=mailbox

   Arguments:
      A <mailbox> (see Section 4.1.2 of [SMTP]) that is associated
      with the identity that submitted the message to the delivery
      system, or the two character sequence "<>" indicating such an
      identity is unknown or insufficiently authenticated. To comply
      with restrictions imposed on ESMTP parameters, the <mailbox> is
      encoded inside an xtext. The syntax of an xtext is described in
      Section 4 of [ESMTP-DSN].

   Note:
      For the purposes of this discussion, "authenticated identity"
      refers to the identity (if any) derived from the authorization
      identity of previous AUTH command, while the terms "authorized
      identity" and "supplied <mailbox>" refer to the sender identity
      that is being associated with a particular message. Note that
      one authenticated identity may be able to identify messages as
      being sent by any number of authorized identities within a
      single session. For example, this may be the case when an SMTP
      server (one authenticated identity) is processing its queue
      (many messages with distinct authorized identities).

   Discussion:
      The optional AUTH parameter to the MAIL FROM command allows
      cooperating agents in a trusted environment to communicate the
      authorization identity associated with individual messages.

      If the server trusts the authenticated identity of the client to
      assert that the message was originally submitted by the supplied
      <mailbox>, then the server SHOULD supply the same <mailbox> in
      an AUTH parameter when relaying the message to any other server
      which supports the AUTH extension.

      For this reason, servers that advertise support for this
      extension MUST support the AUTH parameter to the MAIL FROM
      command even when the client has not authenticated itself to the
      server.

      A MAIL FROM parameter of AUTH=<> indicates that the original
      submitter of the message is not known. The server MUST NOT
      treat the message as having been originally submitted by the
      authenticated identity that resulted from the AUTH command.

If the AUTH parameter to the MAIL FROM command is not supplied, the client has authenticated, and the server believes the message is an original submission, the server MAY generate a <mailbox> from the user's authenticated identity for use in an AUTH parameter when relaying the message to any server which supports the AUTH extension. The generated <mailbox> is implementation specific, but it MUST conform to the syntax of [SMTP]. If the implementation cannot generate a valid <mailbox>, it MUST transmit AUTH=<> when relaying this message.

If the server does not sufficiently trust the authenticated identity of the client, or if the client is not authenticated, then the server MUST behave as if the AUTH=<> parameter was supplied. The server MAY, however, write the value of any supplied AUTH parameter to a log file.

If an AUTH=<> parameter was supplied, either explicitly or due to the requirement in the previous paragraph, then the server MUST supply the AUTH=<> parameter when relaying the message to any server which it has authenticated to using the AUTH extension.

A server MAY treat expansion of a mailing list as a new submission, setting the AUTH parameter to the mailing list address or mailing list administration address when relaying the message to list subscribers.

Note that an implementation which is hard-coded to treat all clients as being insufficiently trusted is compliant with this specification. In that case, the implementation does nothing more than parse and discard syntactically valid AUTH parameters to the MAIL FROM command, and supply AUTH=<> parameters to any servers that it authenticates to.

5.1. Examples

An example where the original identity of the sender is trusted and known:

C: MAIL FROM:<e=mc2@example.com> AUTH=e+3Dmc2@example.com
S: 250 OK

One example where the identity of the sender is not trusted or is otherwise being suppressed by the client:

C: MAIL FROM:<john+@example.org> AUTH=<>
S: 250 OK

6. Status Codes

   The following error codes may be used to indicate various success or
   failure conditions. Servers that return enhanced status codes
   [ESMTP-CODES] SHOULD use the enhanced codes suggested here.

   235 2.7.0 Authentication Succeeded

   This response to the AUTH command indicates that the authentication
   was successful.

   432 4.7.12 A password transition is needed

   This response to the AUTH command indicates that the user needs to
   transition to the selected authentication mechanism. This is
   typically done by authenticating once using the [PLAIN]
   authentication mechanism. The selected mechanism SHOULD then work
   for authentications in subsequent sessions.

   454 4.7.0   Temporary authentication failure

   This response to the AUTH command indicates that the authentication
   failed due to a temporary server failure. The client SHOULD NOT
   prompt the user for another password in this case, and should instead
   notify the user of server failure.

   534 5.7.9   Authentication mechanism is too weak

   This response to the AUTH command indicates that the selected
   authentication mechanism is weaker than server policy permits for
   that user. The client SHOULD retry with a new authentication
   mechanism.

   535 5.7.8   Authentication credentials invalid

   This response to the AUTH command indicates that the authentication
   failed due to invalid or insufficient authentication credentials. In
   this case, the client SHOULD ask the user to supply new credentials
   (such as by presenting a password dialog box).

   500 5.5.6   Authentication Exchange line is too long

   This response to the AUTH command indicates that the authentication
   failed due to the client sending a [BASE64] response that is longer
   than the maximum buffer size available for the currently selected
   SASL mechanism.

530 5.7.0 Authentication required

This response SHOULD be returned by any command other than AUTH,
EHLO, HELO, NOOP, RSET, or QUIT when server policy requires
authentication in order to perform the requested action and
authentication is not currently in force.

538 5.7.11 Encryption required for requested authentication
            mechanism

This response to the AUTH command indicates that the selected
authentication mechanism may only be used when the underlying SMTP
connection is encrypted. Note that this response code is documented
here for historical purposes only. Modern implementations SHOULD NOT
advertise mechanisms that are not permitted due to lack of
encryption, unless an encryption layer of sufficient strength is
currently being employed.

This document adds several new enhanced status codes to the list
defined in [ENHANCED]:

The following 3 Enhanced Status Codes were defined above:

    5.7.8 Authentication credentials invalid
    5.7.9 Authentication mechanism is too weak
    5.7.11 Encryption required for requested authentication mechanism

X.5.6   Authentication Exchange line is too long

This enhanced status code SHOULD be returned when the server fails
the AUTH command due to the client sending a [BASE64] response which
is longer than the maximum buffer size available for the currently
selected SASL mechanism. This is useful for both permanent and
persistent transient errors.

7. Additional Requirements on Servers

As described in Section 4.4 of [SMTP], an SMTP server that receives a
message for delivery or further processing MUST insert the
"Received:" header field at the beginning of the message content.
This document places additional requirements on the content of a
generated "Received:" header field. Upon successful authentication,
a server SHOULD use the "ESMTPA" or the "ESMTPSA" [SMTP-TT] (when
appropriate) keyword in the "with" clause of the Received header
field.

8. Formal Syntax

   The following syntax specification uses the Augmented Backus-Naur
   Form notation as specified in [ABNF]. Non-terminals referenced but
   not defined below are as defined by [ABNF] or [SASL]. The non-
   terminal <mailbox> is defined in [SMTP].

   Except as noted otherwise, all alphabetic characters are case-
   insensitive. The use of upper or lower case characters to define
   token strings is for editorial clarity only. Implementations MUST
   accept these strings in a case-insensitive fashion.

```
     hexchar         = "+" HEXDIG HEXDIG

     xchar           = %x21-2A / %x2C-3C / %x3E-7E
                       ;; US-ASCII except for "+", "=", SP, and CTL

     xtext           = *(xchar / hexchar)
                       ;; non-US-ASCII is only allowed as hexchar

     auth-command    = "AUTH" SP sasl-mech [SP initial-response]
                       *(CRLF [base64]) [CRLF cancel-response]
                       CRLF
                       ;; <sasl-mech> is defined in [SASL]

     auth-param      = "AUTH=" xtext
                       ;; Parameter to the MAIL FROM command.
                       ;; This non-terminal complies with
                       ;; syntax defined by esmtp-param [SMTP].
                       ;;
                       ;; The decoded form of the xtext MUST be
                       ;; either a <mailbox> or the two
                       ;; characters "<>"

     base64          = base64-terminal /
                       ( 1*(4base64-char) [base64-terminal] )

     base64-char     = ALPHA / DIGIT / "+" / "/"
                       ;; Case-sensitive

     base64-terminal = (2base64-char "==") / (3base64-char "=")

     continue-req    = "334" SP [base64] CRLF
                       ;; Intermediate response to the AUTH
                       ;; command.
                       ;; This non-terminal complies with
                       ;; syntax defined by Reply-line [SMTP].
```

        initial-response= base64 / "="

        cancel-response = "*"

9. Security Considerations

    Security issues are discussed throughout this memo.

    If a client uses this extension to get an encrypted tunnel through an
    insecure network to a cooperating server, it needs to be configured
    to never send mail to that server when the connection is not mutually
    authenticated and encrypted. Otherwise, an attacker could steal the
    client's mail by hijacking the [SMTP] connection and either
    pretending the server does not support the Authentication extension
    or causing all AUTH commands to fail.

    Before the [SASL] negotiation has begun, any protocol interactions
    are performed in the clear and may be modified by an active attacker.
    For this reason, clients and servers MUST discard any knowledge
    obtained prior to the start of the SASL negotiation upon the
    establishment of a security layer.

    This mechanism does not protect the TCP port, so an active attacker
    may redirect a relay connection attempt (i.e., a connection between
    two Mail Transfer Agents (MTAs)) to the submission port [SUBMIT].
    The AUTH=<> parameter prevents such an attack from causing a relayed
    message and, in the absence of other envelope authentication, from
    picking up the authentication of the relay client.

    A message submission client may require the user to authenticate
    whenever a suitable [SASL] mechanism is advertised. Therefore, it
    may not be desirable for a submission server [SUBMIT] to advertise a
    SASL mechanism when use of that mechanism grants the clients no
    benefits over anonymous submission.

    Servers MAY implement a policy whereby the connection is dropped
    after a number of failed authentication attempts. If they do so,
    they SHOULD NOT drop the connection until at least 3 attempts to
    authenticate have failed.

    If an implementation supports SASL mechanisms that are vulnerable to
    passive eavesdropping attacks (such as [PLAIN]), then the
    implementation MUST support at least one configuration where these
    SASL mechanisms are not advertised or used without the presence of an
    external security layer such as [TLS].

This extension is not intended to replace or be used instead of end-
to-end message signature and encryption systems such as [S/MIME] or
[PGP]. This extension addresses a different problem than end-to-end
systems; it has the following key differences:

1. It is generally useful only within a trusted enclave.

2. It protects the entire envelope of a message, not just the
    message's body.

3. It authenticates the message submission, not authorship of the
    message content.

4. When mutual authentication is used along with a security layer,
    it can give the sender some assurance that the message was
    successfully delivered to the next hop.

Additional security considerations are mentioned in the [SASL]
specification. Additional security considerations specific to a
particular SASL mechanism are described in the relevant
specification. Additional security considerations for [PLAIN] over
[TLS] are mentioned in Section 15 of this document.

10. IANA Considerations

IANA updated the entry for the "smtp" SASL protocol name to point at
this document.

IANA updated the registration of the Authentication SMTP service
extension as defined in Section 3 of this document. This registry is
currently located at <http://www.iana.org/assignments/mail-
parameters>.

11. Normative References

   [ABNF]   Crocker, D. and P. Overell, "Augmented BNF for Syntax
                 Specifications: ABNF", RFC 4234, October 2005.

   [BASE64]   Josefsson, S., "The Base16, Base32, and Base64 Data
                 Encodings", RFC 4648, October 2006.

   [ESMTP-CODES] Freed, N., "SMTP Service Extension for Returning
                 Enhanced Error Codes", RFC 2034, October 1996.

   [ENHANCED]   Vaudreuil, G., "Enhanced Mail System Status Codes", RFC
                 3463, January 2003.

   [ESMTP-DSN]   Moore, K., "Simple Mail Transfer Protocol (SMTP)
                    Service Extension Delivery Status Notifications
                    (DSNs)", RFC 3461, January 2003.

   [KEYWORDS]   Bradner, S., "Key words for use in RFCs to Indicate
                    Requirement Levels", BCP 14, RFC 2119, March 1997.

   [SASL]   Melnikov, A. and K. Zeilenga, "Simple Authentication
                    and Security Layer (SASL)", RFC 4422, June 2006.

   [SASLprep]   Zeilenga, K., "SASLprep: Stringprep Profile for User
                    Names and Passwords", RFC 4013, February 2005.

   [SMTP]   Klensin, J., "Simple Mail Transfer Protocol", RFC 2821,
                    April 2001.

   [SMTP-TLS]   Hoffman, P., "SMTP Service Extension for Secure SMTP
                    over Transport Layer Security", RFC 3207, February
                    2002.

   [StringPrep]   Hoffman, P. and M. Blanchet, "Preparation of
                    Internationalized Strings ("stringprep")", RFC 3454,
                    December 2002.

   [SUBMIT]   Gellens, R. and J. Klensin, "Message Submission for
                    Mail", RFC 4409, April 2006.

   [SMTP-TT]   Newman, C., "ESMTP and LMTP Transmission Types
                    Registration", RFC 3848, July 2004.

   [PLAIN]   Zeilenga, K., Ed., "The PLAIN Simple Authentication and
                    Security Layer (SASL) Mechanism", RFC 4616, August
                    2006.

   [X509]   Housley, R., Polk, W., Ford, W., and D. Solo, "Internet
                    X.509 Public Key Infrastructure Certificate and
                    Certificate Revocation List (CRL) Profile", RFC 3280,
                    April 2002.

12. Informative References

   [PGP]   Elkins, M., "MIME Security with Pretty Good Privacy
                    (PGP)", RFC 2015, October 1996.

   [S/MIME]   Ramsdell, B., Ed., "Secure/Multipurpose Internet Mail
                    Extensions (S/MIME) Version 3.1 Message Specification",
                    RFC 3851, July 2004.

   [TLS]   Dierks, T. and E. Rescorla, "The Transport Layer
                   Security (TLS) Protocol Version 1.1", RFC 4346, April
                   2006.

   [PIPELINING]   Freed, N., "SMTP Service Extension for Command
                   Pipelining", STD 60, RFC 2920, September 2000.

   [CRAM-MD5]   Klensin, J., Catoe, R., and P. Krumviede, "IMAP/POP
                   AUTHorize Extension for Simple Challenge/Response", RFC
                   2195, September 1997.

13. Acknowledgments

   The editors would like to acknowledge the contributions of John Myers
   and other contributors to RFC 2554, on which this document draws from
   heavily.

   The editors would also like to thank Ken Murchison, Mark Crispin,
   Chris Newman, David Wilson, Dave Cridland, Frank Ellermann, Ned
   Freed, John Klensin, Tony Finch, Abhijit Menon-Sen, Philip Guenther,
   Sam Hartman, Russ Housley, Cullen Jennings, and Lisa Dusseault for
   the time they devoted to reviewing of this document and/or for the
   comments received.

14. Additional Requirements When Using SASL PLAIN over TLS

   This section is normative for SMTP implementations that support SASL
   [PLAIN] over [TLS].

   If an SMTP client is willing to use SASL PLAIN over TLS to
   authenticate to the SMTP server, the client verifies the server
   certificate according to the rules of [X509]. If the server has not
   provided any certificate, or if the certificate verification fails,
   the client MUST NOT attempt to authenticate using the SASL PLAIN
   mechanism.

   After a successful [TLS] negotiation, the client MUST check its
   understanding of the server hostname against the server's identity as
   presented in the server Certificate message, in order to prevent
   man-in-the-middle attacks. If the match fails, the client MUST NOT
   attempt to authenticate using the SASL PLAIN mechanism. Matching is
   performed according to the following rules:

       The client MUST use the server hostname it used to open the
       connection as the value to compare against the server name as
       expressed in the server certificate. The client MUST NOT use

      any form of the server hostname derived from an insecure remote
      source (e.g., insecure DNS lookup). CNAME canonicalization is
      not done.

      If a subjectAltName extension of type dNSName is present in the
      certificate, it SHOULD be used as the source of the server's
      identity.

      Matching is case-insensitive.

      A "*" wildcard character MAY be used as the leftmost name
      component in the certificate. For example, *.example.com would
      match a.example.com, foo.example.com, etc., but would not match
      example.com.

      If the certificate contains multiple names (e.g., more than one
      dNSName field), then a match with any one of the fields is
      considered acceptable.

15. Changes since RFC 2554

   1. Clarified that servers MUST support the use of the AUTH=mailbox
       parameter to MAIL FROM, even when the client is not
       authenticated.

   2. Clarified the initial-client-send requirements, and give
       additional examples.

   3. Updated references to newer versions of various specifications.

   4. Required SASL PLAIN (over TLS) as mandatory-to-implement.

   5. Clarified that the mechanism list can change.

   6. Deprecated the use of the 538 response code.

   7. Added the use of the SASLprep profile for preparing authorization
       identities.

   8. Substantial cleanup of response codes and indicated suggested
       enhanced response codes. Also indicated what response codes
       should result in a client prompting the user for new credentials.

   9. Updated ABNF section to use RFC 4234.

   10. Clarified interaction with SMTP PIPELINING extension.

   11. Added a reference to RFC 3848.

   12. Added a new Enhanced Status Code for "authentication line too
       long" case.

   13. Other general editorial clarifications.

Editors' Addresses

   Robert Siemborski
   Google, Inc.
   1600 Ampitheatre Parkway
   Mountain View, CA 94043, USA

   Phone: +1 650 623 6925
   EMail: robsiemb@google.com

   Alexey Melnikov
   Isode Limited
   5 Castle Business Village, 36 Station Road,
   Hampton, Middlesex, TW12 2BX, UK

   EMail: Alexey.Melnikov@isode.com