Network Working Group   J. Myers
Internet Draft: SMTP Authentication   Carnegie Mellon
Document: draft-myers-smtp-auth-00.txt   April 1995

SMTP Service Extension for Authentication

Status of this Memo

1. Introduction

This document defines an extension to the SMTP service whereby an SMTP client may indicate an authentication mechanism to the server, perform an authentication protocol exchange, and optionally negotiate a protection mechanism for subsequent protocol interactions. The authentication and protection mechanisms used by the the SMTP AUTH extension are those used by the IMAP4 protocol. A mechanism is also provided for a client to transfer envelope authentication of individual messages.

2. The Authentication service extension

(1) the name of the SMTP service extension is "Authentication"

(2) the EHLO keyword value associated with this extension is "AUTH"

(3) no parameter is used with the AUTH EHLO keyword

(4) a new SMTP verb "AUTH" is defined

(5) an optional parameter using the keyword "AUTH" is added to the MAIL FROM command.

3. The AUTH command

AUTH mechanism

Arguments:
a string identifying an IMAP4 authentication mechanism, such as defined by [IMAP4-AUTH]. Any use of the string "imap" used in a server authentication identity in the definition of an authentication mechanism is replaced with the string "smtp".

Restrictions:
after an AUTH command has 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.

Discussion:
The AUTH command indicates an authentication mechanism to the server. If the server supports the requested authentication mechanism, it performs an authentication protocol exchange to authenticate and identify the user. Optionally, it also negotiates a protection mechanism for subsequent protocol interactions. If the requested authentication mechanism is not supported, the server should reject the AUTH command with a 504 reply.

The authentication protocol exchange consists of a series of server challenges and client answers that are specific to the authentication mechanism. A server challenge, otherwise known as a ready response, is a 334 reply with the text part containing a BASE64 encoded string. The client answer consists of a line containing a BASE64 encoded string. If the client wishes to cancel an authentication exchange, it should issue a

line with a single "*". If the server receives such an answer, it must reject the AUTH command by sending a 501 reply.

If the server cannot BASE64 decode the argument, it should reject the AUTH command with a 501 reply. If the server rejects the authentication data, it should reject the AUTH command with a 535 reply. Should the client successfully complete the authentication exchange, the SMTP server issues a 235 reply.

A protection mechanism provides integrity and privacy protection to the protocol session. If a protection mechanism is negotiated, it is applied to all subsequent data sent over the connection. The protection mechanism takes effect immediately following the CRLF that concludes the authentication exchange for the client, and the CRLF of the success reply for the server. Once the protection mechanism is in effect, the stream of command and response octets is processed into buffers of ciphertext. Each buffer is transferred over the connection as a stream of octets prepended with a four octet field in network byte order that represents the length of the following data. The maximum ciphertext buffer length is defined by the protection mechanism.

The server is not required to support any particular authentication mechanism, nor are authentication mechanisms required to support any protection mechanisms. If an AUTH command fails, the client may try another authentication mechanism by issuing another AUTH command. In other words, the client may request authentication types in decreasing order of preference.

The BASE64 string may in general be arbitrarily long. Clients and servers must be able to support challenges and responses that are as long as are generated by the authentication mechanisms they support, independent of any line length

limitations the client or server may have in other parts of its
protocol implementation.

Examples:

```
S: 220 smtp.andrew.cmu.edu ESMTP server ready
C: EHLO jgm.pc.cc.cmu.edu
S: 250-smtp.andrew.cmu.edu
S: 250 AUTH
C: AUTH FOOBAR
S: 504 Unrecognized authentication type
C: AUTH SKEY
S: 334
C: c21pdGg=
S: 334 OTUgUWE1ODMwOA==
C: BsAY3g4gBNo=
S: 235 S/Key authentication successful
```

3. The AUTH parameter to the MAIL FROM command

AUTH=addr-spec

Arguments:
an addr-spec containing the identity which submitted the message to the delivery system. [[length limit? the 64@64 limit of 821 seems a bit small]]

Discussion:
The optional AUTH parameter to the MAIL FROM command allows cooperating agents in a trusted environment to communicate the authentication of individual messages.

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

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 no AUTH parameter was supplied. The server MAY, however, place the value of the AUTH parameter in a comment in the inserted Received: header and/or write it to a log file.

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.

[[encoding spaces or equal signs in the addr-spec?]]

5. Formal Syntax

The following syntax specification uses the augmented Backus-Naur Form (BNF) notation as specified in RFC 822.

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.

```
ATOM_CHAR       ::= <any CHAR except atom_specials>

atom_specials   ::= "(" / ")" / "{" / SPACE / CTLs / "%" / "*" /
                    <"> / "\"
```

```
   auth_command    ::= "AUTH" SPACE auth_type *(CRLF base64) CRLF

   auth_param      ::= "AUTH=" addr-spec
                       ;; addr-spec may not contain SPACE, "="
                       ;; or CTL characters.

   auth_type       ::= 1*ATOM_CHAR

   base64          ::= *(4base64_CHAR) [base64_terminal]

   base64_char     ::= "A" / "B" / "C" / "D" / "E" / "F" / "G" / "H" /
                       "I" / "J" / "K" / "L" / "M" / "N" / "O" / "P" /
                       "Q" / "R" / "S" / "T" / "U" / "V" / "W" / "X" /
                       "Y" / "Z" /
                       "a" / "b" / "c" / "d" / "e" / "f" / "g" / "h" /
                       "i" / "j" / "k" / "l" / "m" / "n" / "o" / "p" /
                       "q" / "r" / "s" / "t" / "u" / "v" / "w" / "x" /
                       "y" / "z" /
                       "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" /
                       "8" / "9" / "+" / "/"
                       ;; Case-sensitive

   base64_terminal ::= (2base64_char "==") / (3base64_char "=")

   CHAR            ::= <any 7-bit US-ASCII character except NUL,
                        0x01 - 0x7f>

   continue_req    ::= "334" SPACE base64 CRLF

   CR              ::= <ASCII CR, carriage return, 0x0C>

   CRLF            ::= CR LF
```

```
   CTL             ::= <any ASCII control character and DEL,
                        0x00 - 0x1f, 0x7f>

   LF              ::= <ASCII LF, line feed, 0x0A>

   SPACE           ::= <ASCII SP, space, 0x20>
```

4. References

[IMAP4-AUTH] Myers, J., "IMAP4 Authentication Mechanisms", RFC 1731, Carnegie Mellon, December 1994.

5. 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.

This extension does not provide a defined mechanism for authentication using a plaintext password. This omission is intentional.

This extension is not intended to replace or be used instead of end-to-end message signature and encryption systems such as PEM 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) it can give the sender some assurance the message was delivered to the next hop

6. Author's Address:

John G. Myers
Carnegie-Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213

EMail: jgm+@cmu.edu