NNTP SASL AUTHentication command

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working
documents of the Internet Engineering Task Force (IETF), its areas,
and its working groups. Note that other groups may also distribute
working documents as Internet-Drafts.

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."

To learn the current status of any Internet-Draft, please check the
``1id-abstracts.txt'' listing contained in the Internet- Drafts
Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe),
munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or
ftp.isi.edu (US West Coast).

1. Introduction

This document describes the optional AUTHSASL command, for indicating an
authentication mechanism to the server, performing an authentication
protocol exchange, and optionally negotiating a protection mechanism
for subsequent protocol interactions. The authentication and
protection mechanisms used by the NNTP AUTHSASL command are those used by
SASL draft.

Please note that large portions of this document are copied from
[POP3-AUTH] with the permission of the author of that document, John G.
Meyers.

2. The AUTHSASL command

AUTHSASL mechanism

Arguments:
   a string identifying an SASL authentication mechanism,
such as defined by [SASL]. If no mechanism is identified
after the AUTHSASL command, it is interpreted as a request
for a list of all mechanisms supported by the server.

Restrictions:
   may only be given in the AUTHORIZATION state
Discussion:
The AUTHSASL 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 AUTHSASL command by sending a negative response.

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 line consisting of a string starting with the 350 continue authorization response code followed by a space, and then followed by a BASE64 encoded string.

The client answer consists solely 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 AUTHSASL command by sending a negative response.

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 positive response 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 AUTHSASL command fails with a negative response, the session remains in the AUTHORIZATION state and client may try another authentication mechanism by issuing another AUTHSASL command, or may attempt to authenticate by using other available authentication commands. In other words, the client may request authentication types in decreasing order of preference, with the AUTH USER command as a last resort.
If a request for a list of all the supported authenticated mechanisms is received, the server will return the list of supported mechanisms, each mechanism on a separate line. The end of the list is denoted by the period character (\').\' on a line by itself. While improbable, it is possible that an implementation of AUTHSASL will return a null list, that is a list consisting of no mechanisms.

Should the client successfully complete the authentication exchange, the NNTP server issues a positive response and the NNTP session enters the EXCHANGE state.

Possible Responses:
215 List of supported mechanisms follows.
250 Authorization accepted
350 Continue with authorization sequence
452 Authorization rejected
501 Command not supported
502 Authentication mechanism not defined.

Examples:
...
C: AUTHSASL
S: KERBEROS_V4
S: .
...
C: AUTHSASL KERBEROS_V4
S: 350 AmFYig==
C: BAcAQUEukVXLknNNVS5FRFUOA0CAsh084kLN3/IJmrMG+25a4DT+nZImJjnTNHJutxAA+o0KPKfHEcAFs9a3CL50ebe/ydHJHwYFdaWwuQ1MWi1y6IesKvjL5rL9WjXUb9MwT9bp0bYLG0Ki1Qh
S: 350 or//EoAADZI=
C: D1AF5A4gA+oOIALuBkAAmw==
S: 250 Kerberos V4 authentication successful
...
C: AUTHSASL KERBEROS_V4
S: 350 AmFYig==
C: *
S: 452 Authorization rejected
...
C: AUTHSASL FOOBAR
S: 502 Unrecognized authentication type

Note: the line breaks in the first client answer are for editorial clarity and are not in real authenticators.
3. 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 ::= "AUTHSASL" 1*(SPACE / TAB) auth_type *(CRLF base64)

auth_type ::= 0*ATOM_CHAR

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

"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" / "+" / "/"

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

CHAR ::= <any 7-bit US-ASCII character except NUL,
0x00 - 0x7f>

continue_req ::= "350" 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


5. Security Considerations

Security issues are discussed throughout this memo.

6. Author's Address

Jack De Winter
Wildbear Consulting, Inc.
17 Brock Street
Kitchener, Ontario, Canada
N2M 1X2

Email: jack@wildbear.on.ca