RFC 4643 - NNTP Extension for Authentication

Network Working Group
Request for Comments: 4643
Updates: 2980
Category: Standards Track

J. Vinocur
Cornell University
K. Murchison
Carnegie Mellon University

October 2006

Network News Transfer Protocol (NNTP)
Extension for Authentication

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This document defines an extension to the Network News Transfer Protocol (NNTP) that allows a client to indicate an authentication mechanism to the server, to perform an authentication protocol exchange, and optionally to negotiate a security layer for subsequent protocol interactions during the remainder of an NNTP session.

This document updates and formalizes the AUTHINFO USER/PASS authentication method specified in RFC 2980 and deprecates the AUTHINFO SIMPLE and AUTHINFO GENERIC authentication methods. Additionally, this document defines a profile of the Simple Authentication and Security Layer (SASL) for NNTP.

Table of Contents

1. Introduction 3
  1.1. Conventions Used in This Document 3
2. The AUTHINFO Extension 4
  2.1. Advertising the AUTHINFO Extension 4

2.2. Authenticating with the AUTHINFO Extension 5
2.3. AUTHINFO USER/PASS Command 6
2.3.1. Usage 7
2.3.2. Description 7
2.3.3. Examples 9
2.4. AUTHINFO SASL Command 9
2.4.1. Usage 10
2.4.2. Description 11
2.4.3. Examples 14
3. Augmented BNF Syntax for the AUTHINFO Extension 16
3.1. Commands 16
3.2. Command Continuation 17
3.3. Responses 17
3.4. Capability Entries 17
3.5. General Non-terminals 18
4. Summary of Response Codes 18
5. Authentication Tracking/Logging 18
6. Security Considerations 19
7. IANA Considerations 20
7.1. IANA Considerations for SASL/GSSAPI Services 20
7.2. IANA Considerations for NNTP Extensions 20
8. Acknowledgements 21
9. References 22
9.1. Normative References 22
9.2. Informative References 22

1. Introduction

Although NNTP [NNTP] has traditionally been used to provide public access to newsgroups, authentication is often useful for several purposes; for example, to control resource consumption, to allow abusers of the POST command to be identified, and to restrict access to "local" newsgroups.

The ad-hoc AUTHINFO USER and AUTHINFO PASS commands, documented in [NNTP-COMMON], provide a very weak authentication mechanism in widespread use by the installed base. Due to their ubiquity, they are formalized in this specification but (because of their insecurity) only for use in combination with appropriate security layers.

The ad hoc AUTHINFO GENERIC command, also documented in [NNTP-COMMON] but much less ubiquitous, provided an NNTP-specific equivalent of the generic SASL [SASL] facility. This document deprecates AUTHINFO GENERIC in favor of an AUTHINFO SASL replacement so that NNTP can benefit from authentication mechanisms developed for other SASL-enabled application protocols, including Simple Mail Transfer Protocol (SMTP) [SMTP-AUTH], Post Office Protocol (POP) [POP-AUTH], Internet Message Access Protocol (IMAP) [IMAP], Lightweight Directory Access Protocol (LDAP) [LDAP-AUTH], and Blocks Extensive Exchange Protocol (BEEP) [BEEP].

This specification is to be read in conjunction with the NNTP base specification [NNTP]. Except where specifically stated otherwise, in the case of a conflict between these two documents, [NNTP] takes precedence over this one.

It is also recommended that this specification be read in conjunction with the SASL base specification [SASL].

1.1. Conventions Used in This Document

The notational conventions used in this document are the same as those in [NNTP], and any term not defined in this document has the same meaning as it does in that one.

The key words "REQUIRED", "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", "MAY", and "OPTIONAL" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels" [KEYWORDS].

Terms related to authentication are defined in "On Internet Authentication" [AUTH].

In the examples, commands from the client are indicated with [C], and responses from the server are indicated with [S].

2. The AUTHINFO Extension

The AUTHINFO extension is used to authenticate a user. Note that authorization is a matter of site policy, not network protocol, and therefore it is not discussed in this document. The server determines authorization in whatever manner is defined by its implementation as configured by the site administrator.

This extension provides three new commands: AUTHINFO USER, AUTHINFO PASS, and AUTHINFO SASL. The capability label for this extension is AUTHINFO.

2.1. Advertising the AUTHINFO Extension

A server MUST implement at least one of the AUTHINFO USER or AUTHINFO SASL commands in order to advertise the "AUTHINFO" capability label in response to the CAPABILITIES command ([NNTP] Section 5.2). However, this capability MUST NOT be advertised after successful authentication (see Section 2.2). This capability MAY be advertised both before and after any use of the MODE READER command ([NNTP] Section 5.3), with the same semantics.

The AUTHINFO capability label contains an argument list detailing which authentication commands are available.

The "USER" argument indicates that AUTHINFO USER/PASS is supported as defined by Section 2.3 of this document. The "USER" argument MUST NOT be advertised, and the AUTHINFO USER/PASS commands SHOULD NOT be provided, unless a strong encryption layer (e.g., Transport Layer Security (TLS) [NNTP-TLS]) is in use or backward compatibility dictates otherwise.

The "SASL" argument indicates that AUTHINFO SASL is supported as defined by Section 2.4 of this document. If the server advertises the "SASL" argument, then it MUST also advertise the "SASL" capability in response to the CAPABILITIES command. The SASL capability is followed by a whitespace-separated list of available SASL mechanism names.

The server MAY list the AUTHINFO capability with no arguments, which indicates that it complies with this specification and does not permit any authentication commands in its current state. In this case, the client MUST NOT attempt to utilize any AUTHINFO commands, even if it contains logic that might otherwise cause it to do so

(e.g., for backward compatibility with servers that are not compliant with this specification).

Future extensions may add additional arguments to this capability. Unrecognized arguments MUST be ignored by the client.

As the AUTHINFO command is related to security, cached results of CAPABILITIES from a previous session MUST NOT be relied on, as per Section 12.6 of [NNTP]. However, a client MAY use such cached results in order to detect active down-negotiation attacks.

Example of AUTHINFO capabilities before and after the use of the STARTTLS [NNTP-TLS] extension:

```
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] IHAVE
[S] STARTTLS
[S] AUTHINFO SASL
[S] SASL CRAM-MD5 DIGEST-MD5 GSSAPI
[S] LIST ACTIVE NEWSGROUPS
[S] .
[C] STARTTLS
[S] 382 Continue with TLS negotiation
[TLS negotiation proceeds, further commands protected by TLS]
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] IHAVE
[S] AUTHINFO USER SASL
```

[S] SASL CRAM-MD5 DIGEST-MD5 GSSAPI PLAIN EXTERNAL
[S] LIST ACTIVE NEWSGROUPS
[S] .

2.2. Authenticating with the AUTHINFO Extension

An NNTP server responds to a client command with a 480 response to indicate that the client MUST authenticate and/or authorize in order to use that command or access the indicated resource. Use of the AUTHINFO command as described below is one such way that a client can authenticate/authorize to the server. The client MAY therefore use an AUTHINFO command after receiving a 480 response. A client intending to use an AUTHINFO command SHOULD issue the CAPABILITIES command to obtain the available authentication commands and mechanisms before attempting authentication.

If a server advertises the AUTHINFO capability, a client MAY attempt the first step of authentication at any time during a session to acquire additional privileges without having received a 480 response. Servers SHOULD accept such unsolicited authentication requests. A server MUST NOT under any circumstances reply to an AUTHINFO command with a 480 response.

A client MUST NOT under any circumstances continue with any steps of authentication beyond the first, unless the response code from the server indicates that the authentication exchange is welcomed. In particular, anything other than a 38x response code indicates that the client MUST NOT continue the authentication exchange.

After a successful authentication, the client MUST NOT issue another AUTHINFO command in the same session. A server MUST NOT return the AUTHINFO capability in response to a CAPABILITIES command, and a server MUST reject any subsequent AUTHINFO commands with a 502 response. Additionally, the client MUST NOT issue a MODE READER command after authentication, and a server MUST NOT advertise the MODE-READER capability.

In agreement with [SASL], the server MUST continue to advertise the SASL capability in response to a CAPABILITIES command with the same list of SASL mechanisms that it did before authentication (thereby enabling the client to detect a possible active down-negotiation attack). Other capabilities returned in response to a CAPABILITIES command received after authentication MAY be different from those returned before authentication. For example, an NNTP server may not want to advertise support for a specific extension unless a client has been authenticated.

Note that a server may perform a successful authentication exchange with a client and yet still deny access to some or all resources; the permanent 502 response indicates that a resource is unavailable even though authentication has been performed (this is in contrast to the temporary 480 error, which indicates that a resource is unavailable now but may become available after authentication).

2.3. AUTHINFO USER/PASS Command

This section supersedes the definition of the AUTHINFO USER and AUTHINFO PASS commands as documented in Section 3.1.1 of

[NNTP-COMMON].

2.3.1. Usage

These commands MUST NOT be pipelined.

Syntax
AUTHINFO USER username
AUTHINFO PASS password

Responses
281 Authentication accepted
381 Password required [1]
481 Authentication failed/rejected
482 Authentication commands issued out of sequence
502 Command unavailable [2]

[1] Only valid for AUTHINFO USER. Note that unlike traditional 3xx codes, which indicate that the client may continue the current command, the legacy 381 code means that the AUTHINFO PASS command must be used to complete the authentication exchange.

[2] If authentication has already occurred, AUTHINFO USER/PASS are not valid commands (see Section 2.2).

NOTE: Notwithstanding Section 3.2.1 of [NNTP], the server MUST NOT return 480 in response to AUTHINFO USER/PASS.

Parameters
username = string identifying the user/client
password = string representing the user's password

2.3.2. Description

The AUTHINFO USER and AUTHINFO PASS commands are used to present clear text credentials to the server. These credentials consist of a username or a username plus a password (the distinction is that a password is expected to be kept secret, whereas a username is not; this does not directly affect the protocol but may have an impact on user interfaces). The username is supplied through the AUTHINFO USER command, and the password through the AUTHINFO PASS command.

If the server requires only a username, it MUST NOT give a 381 response to AUTHINFO USER and MUST give a 482 response to AUTHINFO PASS.

If the server requires both username and password, the former MUST be sent before the latter. The server will need to cache the username until the password is received; it MAY require that the password be

sent in the immediately next command (in other words, only caching the username until the next command is sent). The server:

- MUST return a 381 response to AUTHINFO USER;

- MUST return a 482 response to AUTHINFO PASS if there is no cached username;

- MUST use the argument of the most recent AUTHINFO USER for authentication; and

- MUST NOT return a 381 response to AUTHINFO PASS.

The server MAY determine whether a password is needed for a given username. Thus the same server can respond with both 381 and other response codes to AUTHINFO USER.

Should the client successfully present proper credentials, the server issues a 281 reply. If the server is unable to authenticate the client, it MUST reject the AUTHINFO USER/PASS command with a 481 reply. If an AUTHINFO USER/PASS command fails, the client MAY proceed without authentication. Alternatively, the client MAY try another authentication mechanism or present different credentials by issuing another AUTHINFO command.

The AUTHINFO PASS command permits the client to use a clear-text password to authenticate. A compliant implementation MUST NOT implement this command without also implementing support for TLS [NNTP-TLS]. Use of this command without an active strong encryption layer is deprecated, as it exposes the user's password to all parties on the network between the client and the server. Any implementation of this command SHOULD be configurable to disable it whenever a strong encryption layer (such as that provided by [NNTP-TLS]) is not active, and this configuration SHOULD be the default. The server will use the 483 response code to indicate that the datastream is insufficiently secure for the command being attempted (see Section 3.2.1 of [NNTP]).

Note that a server MAY (but is not required to) allow white space characters in usernames and passwords. A server implementation MAY blindly split command arguments at white space and therefore may not preserve the exact sequence of white space characters in the username or password. Therefore, a client SHOULD scan the username and password for white space and, if any is detected, warn the user of the likelihood of problems. The SASL PLAIN [PLAIN] mechanism is recommended as an alternative, as it does not suffer from these issues.

Also note that historically the username is not canonicalized in any way. Servers MAY use the [SASLprep] profile of the [StringPrep] algorithm to prepare usernames for comparison, but doing so may cause interoperability problems with legacy implementations. If

canonicalization is desired, the SASL PLAIN [PLAIN] mechanism is recommended as an alternative.

2.3.3. Examples

Example of successful AUTHINFO USER:

```
[C] AUTHINFO USER wilma
[S] 281 Authentication accepted
```

Example of successful AUTHINFO USER/PASS:

```
[C] AUTHINFO USER fred
[S] 381 Enter passphrase
[C] AUTHINFO PASS flintstone
[S] 281 Authentication accepted
```

Example of AUTHINFO USER/PASS requiring a security layer:

```
[C] AUTHINFO USER fred@stonecanyon.example.com
[S] 483 Encryption or stronger authentication required
```

Example of failed AUTHINFO USER/PASS:

```
[C] AUTHINFO USER barney
[S] 381 Enter passphrase
[C] AUTHINFO PASS flintstone
[S] 481 Authentication failed
```

Example of AUTHINFO PASS before AUTHINFO USER:

```
[C] AUTHINFO PASS flintstone
[S] 482 Authentication commands issued out of sequence
```

2.4. AUTHINFO SASL Command

This section defines a formal profile of the Simple Authentication and Security Layer [SASL]. The use of the AUTHINFO GENERIC command as documented in Section 3.1.3 of [NNTP-COMMON], as a way to perform SASL authentication, is deprecated in favor of the AUTHINFO SASL command. A server SHOULD NOT advertise AUTHINFO GENERIC in the list of capabilities returned by CAPABILITIES.

2.4.1. Usage

This command MUST NOT be pipelined.

Syntax

```
AUTHINFO SASL mechanism [initial-response]
```

This command MAY exceed 512 octets. The maximum length of this command is increased to that which can accommodate the largest encoded initial response possible for any of the SASL mechanisms supported by the implementation.

Responses

```
281          Authentication accepted
283 challenge  Authentication accepted (with success data) [1]
383 challenge  Continue with SASL exchange [1]
481          Authentication failed/rejected
482          SASL protocol error
502          Command unavailable [2]

[1] These responses MAY exceed 512 octets.  The maximum length of
    these responses is increased to that which can accommodate the
    largest encoded challenge possible for any of the SASL
    mechanisms supported by the implementation.

[2] If authentication has already occurred, AUTHINFO SASL is not a
    valid command (see Section 2.2).

    NOTE: Notwithstanding Section 3.2.1 of [NNTP], the server MUST
    NOT return 480 in response to AUTHINFO SASL.

Parameters
  mechanism        = String identifying a [SASL] authentication
                     mechanism.
  initial-response  = Optional initial client response.
                     If present, the response MUST be encoded as
                     specified in Section 4 of [BASE64]. [3]
  challenge        = Server challenge.
                     The challenge MUST be encoded as specified
                     in Section 4 of [BASE64].

[3] This argument MAY exceed 497 octets.  The maximum length of
    this argument is increased to that which can accommodate the
    largest encoded initial response possible for any of the SASL
    mechanisms supported by the implementation.
```


2.4.2. Description

The AUTHINFO SASL command initiates a [SASL] exchange between the client and the server. The client identifies the SASL mechanism to be used with the first parameter of the AUTHINFO SASL 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), the server rejects the AUTHINFO SASL command with a 503 reply (see Section 3.2.1 of [NNTP]). If the requested authentication mechanism requires an encryption layer, the server rejects the AUTHINFO SASL command with a 483 reply (see Section 3.2.1 of [NNTP]).

The service name specified by this protocol's profile of SASL is "nntp".

The SASL 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 383 reply with a single argument

containing the [BASE64]-encoded string supplied by the SASL mechanism. A server challenge that has zero length MUST be sent as a single equals sign ("=") and MUST be included (in order to comply with the [NNTP] requirement that responses always have the same number of arguments).

A client response consists of a line containing a [BASE64]-encoded string. A client response that has zero length MUST be sent as a single equals sign ("=") and MUST be included (for consistency with the server challenge format). 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 AUTHINFO SASL command by sending a 481 reply.

Note that these [BASE64]-encoded strings can be much longer than normal NNTP responses. 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.

The optional initial response argument to the AUTHINFO SASL 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 NNTP, a server challenge that contains no data is equivalent to a zero-length challenge and is encoded as a single equals sign ("=").

Note that the [BASE64]-encoded initial response argument can exceed 497 octets, and therefore that the AUTHINFO SASL command can exceed 512 octets. Clients SHOULD and servers MUST be able to handle the maximum encoded size of initial responses possible for their supported authentication mechanisms. This requirement is independent of any command or argument length limitations the client or server may have in other parts of its protocol implementation.

If use of the initial response argument would cause the AUTHINFO SASL command to exceed 512 octets, the client MAY choose to omit 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 that it contains no data.

If the client uses an initial-response argument to the AUTHINFO SASL command with a SASL mechanism that does not support an initial client response, the server MUST reject the AUTHINFO SASL command with a 482 reply.

If the server cannot [BASE64] decode any client response, it MUST reject the AUTHINFO SASL command with a 504 reply (see Section 3.2.1 of [NNTP]). 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 they 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).

The authorization identity generated by this [SASL] exchange is a simple username, and both client and server MUST 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 with a 481 reply.

Should the client successfully complete the exchange, the server issues either a 281 or a 283 reply. If the server is unable to authenticate the client, it MUST reject the AUTHINFO SASL command

with a 481 reply. If an AUTHINFO SASL command fails, the client MAY proceed without authentication. Alternatively, the client MAY try another authentication mechanism, or present different credentials by issuing another AUTHINFO command.

If the SASL mechanism returns additional data on success (e.g., server authentication), the NNTP server issues a 283 reply with a single argument containing the [BASE64]-encoded string supplied by the SASL mechanism. If no additional data is returned on success, the server issues a 281 reply.

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 NNTP protocol is reset to the state immediately after the initial greeting response (see 5.1 of [NNTP]) has been sent, with the exception that if a MODE READER command has been issued, the effects of it (if any) are not reversed. The server MUST discard any knowledge obtained from the client, such as the current newsgroup and article number, that was not obtained from the SASL negotiation itself. Likewise, the client SHOULD discard and MUST NOT rely on any knowledge obtained from the server, such as the capability list, that 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.)

When both TLS [NNTP-TLS] and SASL security layers are in effect, the TLS encoding MUST be applied after the SASL encoding (the cleartext data is always SASL encoded first, and then the resultant data is TLS encoded).

To ensure interoperability, client and server implementations of this extension MUST implement the [DIGEST-MD5] SASL mechanism.

If AUTHINFO USER/PASS and AUTHINFO SASL are both implemented, the SASL [PLAIN] mechanism SHOULD also be implemented, as the

functionality of DIGEST-MD5 is insufficient for some environments (e.g., the server may need to pass off the plaintext password to an external authentication service). The SASL PLAIN mechanism is preferred over AUTHINFO USER, even if there is not a strong encryption layer active, because it eliminates limitations that AUTHINFO USER/PASS has with regards to the use of white space characters being used in usernames and passwords.

2.4.3. Examples

Example of the [PLAIN] SASL mechanism under a TLS layer, using an initial client response:

```
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] STARTTLS
[S] AUTHINFO SASL
[S] SASL CRAM-MD5 DIGEST-MD5 GSSAPI
[S] LIST ACTIVE NEWSGROUPS
[S] .
[C] STARTTLS
[S] 382 Continue with TLS negotiation
[TLS negotiation proceeds, further commands protected by TLS]
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] AUTHINFO USER SASL
[S] SASL CRAM-MD5 DIGEST-MD5 GSSAPI PLAIN EXTERNAL
[S] LIST ACTIVE NEWSGROUPS
[S] .
[C] AUTHINFO SASL PLAIN AHRlc3QAMTIzNA==
[S] 281 Authentication accepted
```

Example of the EXTERNAL SASL mechanism under a TLS layer, using the authorization identity derived from the client TLS certificate, and thus a zero-length initial client response (commands prior to AUTHINFO SASL are the same as the previous example and have been omitted):

```
[C] AUTHINFO SASL EXTERNAL =
[S] 281 Authentication accepted
```

Example of the [DIGEST-MD5] SASL mechanism, which includes a server challenge and server success data (white space has been inserted for clarity; base64-encoded data is actually sent as a single line with no embedded white space):

```
[C] AUTHINFO SASL DIGEST-MD5
[S] 383 bm9uY2U9InNheUFPaENFS0dJZFBNSEMwd3RsZUxxT0ljT0kyd1FZSWU0
    enplQXR1aVE9IixyZWFsbT0iZWFnbGUub2NlYW5hLmNvbSIscW9wPSJhdXRo
    LGF1dGgtaW50LGF1dGgtY29uZiIsY2lwaGVyPSJyYzQtNDAscmM0LTU2LHJj
    NCxkZXMsM2RlcyIsbWF4YnVmPTQwOTYsY2hhcnNldD11dGYtOCxhbGdvcml0
    aG09bWQ1LXNlc3M=
```

[C] dXNlcm5hbWU9InRlc3QiLHJlYWxtPSJlYWdsZS5vY2VhbmEuY29tIixub25j
ZT0ic2F5QU9oQ0VLR0lkUE1IQzB3dGxlTHFPSWNPSTJ3UVlJZTR6emVBdHVp
UT0iLGNub25jZT0iMFkzSlFWMlRnOVNjRGlwK08xU1ZDMHJoVmcvLytkbk9J
aUd6LzdDZU5KOD0iLG5jPTAwMDAwMDAxLHFvcD1hdXRoLWNvbmYsY2lwaGVy
PXJjNCxtYXhidWY9MTAyNCxkaWdlc3QtdXJpPSJubnRwL2xvY2FsaG9zdCIs
cmVzcG9uc2U9ZDQzY2Y2NmNmZmE5MDNmOWViMDM1NmMwOGEzZGIwZjI=
[S] 283 cnNwYXV0aD1kZTJlMTI3ZTVhODFjZGE1M2Q5N2FjZGEzNWNkZTgzYQ==

Example of a failed authentication due to bad [GSSAPI] credentials. Note that although the mechanism can utilize the initial response, the client chooses not to use it because of its length, resulting in a zero-length server challenge (here, white space has been inserted for clarity; base64-encoded data is actually sent as a single line with no embedded white space):

[C] AUTHINFO SASL GSSAPI
[S] 383 =
[C] YIICOAYJKoZIhvcSAQICAQBuggInMIICI6ADAgEFoQMCAQ6iBwMFACAAAACj
ggE/YYIBOzCCATegAwIBBaEYGxZURVNULk5FVC5JU0MuVVBFTk4uRURVoiQw
IqADAgEDoRswGRsEbmV3cxsRbmV0bmV3cy51cGVubi5lZHWjge8wgeygAwIB
EKEDAgECooHfBIHcSQfLKC8vm2i17EXmomwk6hHvjBY/BnKnvvDTrbno3198
vlX2RSUt+CjuAKhcDcj4DW0gvZEqH7t5v9yWedzztlpaThebBat6hQNr9NJP
ozh1/+74HUwhGWb50KtjuftO/ftQ8q0nTuYKgIq6PM4tp2ddo1IfpjfdNR9E
95GFi3y1uBT7lQOwtQbRJUjPSO3ijdue9V7cNNVmYsBsqNsaHhvlBJEXf4WJ
djH8yG+Dw/gX8fUTtC5fDpB5zLt01mkSXh6Wc4UhqQtwZBI2t/+TpX1okbg6
Hr1ZZupeH6SByjCBx6ADAgEQooG/BIG8GnCmcXWtqhXh48dGTLHQgJ04K5Fj
RMMq2qPSbiha9lq0osqR2KAnQA6LioWYxU+6yPKpBDSC5WOT441fUfkM8iAL
kW3uNc+luFCGcnDsacrmoVU7Y6Akcp9m7Fm7orRc+TWSWPpBg3OR2oG3ATW0
0NAz8TT06VOLVxIMUTINKdYVI/Ja7f3sy+/N4LGkJqScCQOwlo5tfDWn/UQF
iTWo5Zw435rH8pjy2smQCnqC14v3NMAWTu4j+dzHUNw=
[S] 481 Authentication error

Example of a client aborting in the midst of an exchange:

[C] AUTHINFO SASL GSSAPI
[S] 383 =
[C] *
[S] 481 Authentication aborted as requested

Example of attempting to use a mechanism that is not supported by the server:

[C] AUTHINFO SASL EXAMPLE
[S] 503 Mechanism not recognized

Example of attempting to use a mechanism that requires a security layer:

```
[C] AUTHINFO SASL PLAIN
[S] 483 Encryption or stronger authentication required
```

Example of using an initial response with a mechanism that doesn't support it (the server must start the exchange when using [CRAM-MD5]):

```
[C] AUTHINFO SASL CRAM-MD5 AHRlc3QAMTIzNA==
[S] 482 SASL protocol error
```

Example of an authentication that failed due to an incorrectly encoded response:

```
[C] AUTHINFO SASL CRAM-MD5
[S] 383 PDE1NDE2NzQ5My4zMjY4MzE3QHRlc3RAZXhhbXBsZS5jb20+
[C] abcd=efg
[S] 504 Base64 encoding error
```

3. Augmented BNF Syntax for the AUTHINFO Extension

This section describes the formal syntax of the AUTHINFO extension using ABNF [ABNF]. It extends the syntax in Section 9 of [NNTP], and non-terminals not defined in this document are defined there. The [NNTP] ABNF should be imported first before attempting to validate these rules.

3.1. Commands

This syntax extends the non-terminal "command", which represents an NNTP command.

```
command =/ authinfo-sasl-command /
    authinfo-user-command /
    authinfo-pass-command

authinfo-sasl-command = "AUTHINFO" WS "SASL" WS mechanism
    [WS initial-response]
authinfo-user-command = "AUTHINFO" WS "USER" WS username
authinfo-pass-command = "AUTHINFO" WS "PASS" WS password

initial-response = base64-opt
username = 1*user-pass-char
password = 1*user-pass-char
user-pass-char = B-CHAR
```


NOTE: a server implementation MAY parse AUTHINFO USER and AUTHINFO PASS specially so as to allow white space to be used within the username or password. Such implementations accept the additional syntax (making these two items inconsistent with "token" in Section 9.8 of [NNTP]):

```
user-pass-char =/ SP / TAB
```

In doing so, the grammar can become ambiguous if the username or password begins or ends with white space. To solve this ambiguity, such implementations typically treat everything after the first white

space character following "USER"/"PASS", up to, but not including, the CRLF, as the username/password.

3.2. Command Continuation

This syntax extends the non-terminal "command-continuation", which represents the further material sent by the client in the case of multi-stage commands.

```
command-continuation =/ authinfo-sasl-383-continuation

authinfo-sasl-383-continuation = ("*" / base64-opt) CRLF
```

3.3. Responses

This syntax extends the non-terminal "initial-response-content", which represents an initial response line sent by the server.

```
initial-response-content =/ response-283-content /
    response-383-content

response-283-content = "283" SP base64
response-383-content = "383" SP base64-opt
```

3.4. Capability Entries

This syntax extends the non-terminal "capability-entry", which represents a capability that may be advertised by the server.

```
capability-entry =/ authinfo-capability /
    sasl-capability

authinfo-capability = "AUTHINFO" *(WS authinfo-variant)
authinfo-variant = "USER" / "SASL"
sasl-capability = "SASL" 1*(WS mechanism)
```


3.5. General Non-terminals

```
base64-opt = "=" / base64
mechanism = 1*20mech-char
mech-char = UPPER / DIGIT / "-" / "_"
```

4. Summary of Response Codes

This section contains a list of each new response code defined in this document and indicates whether it is multi-line, which commands can generate it, what arguments it has, and what its meaning is.

Response code 281
  Generated by: AUTHINFO USER, AUTHINFO PASS, AUTHINFO SASL
  Meaning: authentication accepted

Response code 283
  Generated by: AUTHINFO SASL
  1 argument: challenge
  Meaning: authentication accepted (with success data)

Response code 381
Generated by: AUTHINFO USER
Meaning: password required via AUTHINFO PASS command. Note that this code is used for backwards compatibility and does not conform to the traditional use of 3xx codes.

Response code 383
Generated by: AUTHINFO SASL
1 argument: challenge
Meaning: continue with SASL exchange

Response code 481
Generated by: AUTHINFO USER, AUTHINFO PASS, AUTHINFO SASL
Meaning: authentication failed/rejected

Response code 482
Generated by: AUTHINFO USER, AUTHINFO PASS, AUTHINFO SASL
Meaning: authentication commands issued out of sequence or SASL protocol error

5. Authentication Tracking/Logging

This section contains implementation suggestions and notes of best current practice; it does not specify further network protocol requirements.

Once authenticated, the authorization identity presented in the AUTHINFO exchange (username when using USER/PASS) SHOULD be included in an audit trail associating the identity with any articles supplied during a POST operation, and this configuration SHOULD be the default. This may be accomplished, for example, by inserting headers in the posted articles or by a server logging mechanism. The server MAY provide a facility for disabling the procedure described above, as some users or administrators may consider it a violation of privacy.

6. Security Considerations

Security issues are discussed throughout this memo.

In general, the security considerations of [SASL] and any implemented SASL mechanisms are applicable here; only the most important are highlighted specifically below. Also, this extension is not intended to cure the security considerations described in section 12 of [NNTP]; those considerations remain relevant to any NNTP implementation.

Before the [SASL] negotiation has begun, any protocol interactions may have been performed in the clear and may have been modified by an active attacker. For this reason, clients and servers MUST discard any sensitive knowledge obtained prior to the start of the SASL negotiation upon the establishment of a security layer. Furthermore, the CAPABILITIES command SHOULD be re-issued upon the establishment of a security layer, and other protocol state SHOULD be re-negotiated as well.

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 at authentication have failed.

Implementations MUST support a configuration where authentication mechanisms that are vulnerable to passive eavesdropping attacks (such as AUTHINFO USER/PASS and SASL [PLAIN]) are not advertised or used without the presence of an external security layer such as TLS [NNTP-TLS], and this configuration SHOULD be the default.

When multiple authentication mechanisms are permitted by both client and server, an active attacker can cause a down-negotiation to the weakest mechanism. For this reason, both clients and servers SHOULD be configurable to forbid use of weak mechanisms. The minimum strength acceptable is a policy decision that is outside the scope of this specification.

7. IANA Considerations

7.1. IANA Considerations for SASL/GSSAPI Services

The IANA has registered the SASL/GSSAPI service name "nntp". This service name refers to authenticated use of Usenet news service when it is provided via the [NNTP] protocol.

- o Published Specification: This document.
- o Contact for Further Information: Authors of this document.
- o Change Controller: IESG <iesg@ietf.org>.

7.2. IANA Considerations for NNTP Extensions

This section gives a formal definition of the AUTHINFO extension, as required by Section 3.3.3 of [NNTP] for the IANA registry.

- o This extension provides an extensible mechanism for NNTP authentication via a variety of methods.
- o The capability label for this extension is "AUTHINFO".
- o The "AUTHINFO" capability label has two possible optional arguments, "USER" and "SASL" (as defined in Section 2.1), indicating which variants of the AUTHINFO command are supported.
- o This extension also provides the "SASL" capability label, whose arguments list the available SASL mechanisms.
- o This extension defines three new commands, AUTHINFO USER, AUTHINFO PASS, and AUTHINFO SASL, whose behavior, arguments, and responses are defined in Sections 2.3 and 2.4.
- o This extension does not associate any new responses with pre-existing NNTP commands.

- o This extension may affect the overall behavior of both server and client in that the AUTHINFO SASL command may require that subsequent communication be transmitted via an intermediary security layer.

- o The length of the AUTHINFO SASL command (as defined in this document) may exceed 512 octets. The maximum length of this command is increased to that which can accommodate the largest initial response possible for any of the SASL mechanisms supported by the implementation.

- o This extension defines two new responses, 283 and 383, whose lengths may exceed 512 octets. The maximum length of these responses is increased to that which can accommodate the largest challenge possible for any of the SASL mechanisms supported by the implementation.

- o This extension does not alter pipelining, but AUTHINFO commands cannot be pipelined.

- o Use of this extension may alter the capabilities list; once the AUTHINFO command has been used successfully, the AUTHINFO capability can no longer be advertised by CAPABILITIES. Additionally, the MODE-READER capability MUST NOT be advertised after successful authentication.

- o This extension does not cause any pre-existing command to produce a 401, 480, or 483 response.

- o This extension is unaffected by any use of the MODE READER command; however, the MODE READER command MUST NOT be used in the same session following successful authentication.

- o Published Specification: This document.

- o Contact for Further Information: Authors of this document.

- o Change Controller: IESG <iesg@ietf.org>.

8. Acknowledgements

This RFC originated from a document initially written by Chris Newman.

A significant amount of the authentication text was originally from the NNTP revision or common authentication specs written by Stan Barber. A significant amount of the SASL text was lifted from the revisions to RFC 1734 and RFC 2554 by Rob Siemborski.

Special acknowledgement also goes to Russ Allbery, Clive Feather, and others who commented privately on intermediate revisions of this document, as well as the members of the IETF NNTP Working Group for continual (yet sporadic) insight in discussion.

9. References

9.1. Normative References

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

[AUTH] Haller, N. and R. Atkinson, "On Internet Authentication", RFC 1704, October 1994.

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

[DIGEST-MD5] Leach, P. and C. Newman, "Using Digest Authentication as a SASL Mechanism", RFC 2831, May 2000.

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

[NNTP] Feather, C., "Network News Transfer Protocol (NNTP)", RFC 3977, October 2006.

[NNTP-TLS] Murchison, K., Vinocur, J., and C. Newman, "Using Transport Layer Security (TLS) with Network News Transfer Protocol (NNTP)", RFC 4642, October 2006.

[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.

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

9.2. Informative References

[BEEP] Rose, M., "The Blocks Extensible Exchange Protocol Core", RFC 3080, March 2001.

[CRAM-MD5] Nerenberg, L., "The CRAM-MD5 SASL Mechanism", Work in Progress.

[GSSAPI] Melnikov, A., "SASL GSSAPI mechanisms", Work in Progress.

[IMAP] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

[LDAP-AUTH] Harrison, R., "Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms", RFC 4513, June 2006.

[NNTP-COMMON] Barber, S., "Common NNTP Extensions", RFC 2980, October 2000.

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

[POP-AUTH] Myers, J., "POP3 AUTHentication command", RFC 1734, December 1994.

[SMTP-AUTH] Myers, J., "SMTP Service Extension for Authentication", RFC 2554, March 1999.

Authors' Addresses

Jeffrey M. Vinocur
Department of Computer Science
Upson Hall
Cornell University
Ithaca, NY 14853 USA

EMail: vinocur@cs.cornell.edu

Kenneth Murchison
Carnegie Mellon University
5000 Forbes Avenue
Cyert Hall 285
Pittsburgh, PA 15213 USA

EMail: murch@andrew.cmu.edu

Full Copyright Statement

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an

"AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).