Network Working Group   Robert Siemborski
INTERNET-DRAFT   Carnegie Mellon University
Intended Category: Proposed Standard   December, 2003

POP3 SASL Authentication Mechanism

<draft-siemborski-rfc1734bis-02.txt>

Abstract

This document defines a profile of the Simple Authentication and Security Layer (SASL) for the Post Office Protocol (POP3). This extension allows a POP3 client to 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.

In order to consolidate all of the authentication related information for POP3 into a single document, this document obsoletes RFC 1734 and RFC 3206, replacing them as a Proposed Standard. It also updates information contained in Section 6.3 and Section 8 of RFC 2449.

Table of Contents

1. How to Read This Document

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

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

2. Introduction

The [POP3] AUTH command [POP3-AUTH] in has suffered several problems in its specification. The first is that it was very similar to a [SASL] framework, but pre-dated the initial SASL specification. It was therefore missing some key components, such as a way to list the available authentication mechanisms.

Later, [POP3-EXT] attempted to remedy this situation by adding the CAPA command and allowing an initial client response to the AUTH command, however problems in the clarity of the specification of how the initial client response was to be handled remained.

Additionally, there is yet another document, [POP3-CODES], that provides additional response codes that are useful during authentication. Together, this means creating a full POP3 AUTH implementaiton requires an understanding of material in atleast six different documents.

This document attempts to combine all of the POP3 SASL authentication related details into a single document, in addition to clarifying and updating the older specifications where appropriate.

3. The SASL Capability

This section supercedes the definition of the SASL Capability in section 6.3 of [POP3-EXT].

CAPA tag:
    SASL

Arguments:
    Supported SASL Mechanisms

Standard Commands Affected
    None

Announced states / possible differences:
    both / no

Commands valid in states:
    AUTHORIZATION

Specification Reference:
    This Document, [SASL]

Discussion
    The SASL capability permits the use of the AUTH command (as defined in section 4 of this document) to begin a [SASL] negotiation. The arguments to the SASL capability is a space-separated list of SASL mechanisms which are supported.

    If a server either does not support the CAPA command or does not advertise the SASL capability, clients SHOULD NOT attempt the AUTH command. If a client does attempt the AUTH command in such a situation, it MUST NOT supply the client initial response parameter (for backwards compatibility with [POP3-AUTH]).

    Note that the list of available mechanisms MAY change after a successful STLS command [POP3-TLS]. Additionally, implementations MAY choose to omit the SASL capability after a successful AUTH command has been completed.

Example

```
    S: +OK pop.example.com BlurdyBlurp POP3 server ready
    C: CAPA
    S: +OK List of capabilities follows
    S: SASL KERBEROS_V4 GSSAPI ANONYMOUS
    S: STLS
    S: IMPLEMENTATION BlurdyBlurp POP3 server
    S: .
```

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 specified in Section 3 of [BASE64].

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 -ERR reply.

The AUTH command may only be given during the AUTHORIZATION state.

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 not supported, the server rejects the AUTH command with a -ERR reply.

The authentication protocol 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 line consisting of a "+" character followed by a single space and a string encoded as specified in Section 3 of [BASE64]. This challenge MUST NOT contain any text other than the BASE64 encoded challenge.

A client response consists of a line containing a string encoded as defined in Section 3 of [BASE64]. 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 -ERR 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 POP3, a server challenge with no data is defined as line with only a "+" followed by a single space. It MUST NOT contain any other data.

For the purposes of the initial client response, the line length limitation defined in [POP3-EXT] still applies. If a client initial send 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 needs to send a zero-length initial response, the client MUST 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 that does not support an initial client send, the server MUST reject the AUTH command with a -ERR reply.

If the server cannot [BASE64] decode any client response, it MUST reject the AUTH command with a -ERR reply. 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 (excepting the initial client response) may be of arbitrarily length. 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.

If the server is unable to authenticate the client, it MUST reject the AUTH command with a -ERR reply. Should the client successfully complete the exchange, the server issues a +OK reply. Additionally, upon success, the POP3 session enters the TRANSACTION state.

The authorization identity generated by this [SASL] exchange is a simple username, and MUST use the [SASLprep] profile of the [StringPrep] algorithm to prepare these names for matching. 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.

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 server MUST discard any knowledge previously obtained from the client, 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 available POP3 service extensions. After a security layer is established, the server SHOULD NOT advertise either the SASL or the STLS extension.

When 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 "pop".

If an AUTH command fails, the client may try another authentication mechanism or present different credentials by issuing another AUTH command (or by using one of the other [POP3] authentication mechanisms). Likewise, the server MUST behave as if the client had not issued the AUTH command.

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

<<Open Issue: Is this the best choice of mandatory-to-implement mechanism for POP3? IMAP arrived at a choice that equates to STLS+PLAIN, and therefore is likely to be implemented in clients already. Is there really a compelling reason to choose something else?

DIGEST-MD5 has been suggested as a choice that does not require servers to implement TLS, which is desireable from a code complexity/deployability standpoint. However, DIGEST-MD5 also requires the storage of (essentially) plaintext equivilent passwords which also may not be acceptable in some enviornments.>>

4.1. Formal Syntax

The following syntax specification uses the Augmented Backus-Naur Form notation as specified in [ABNF].

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.

```
    UPALPHA         = %x41-5A            ;; Uppercase: A-Z

    LOALPHA         = %x61-7A            ;; Lowercase: a-z

    ALPHA           = UPALPHA / LOALPHA  ;; case insensitive

    DIGIT           = %x30-39            ;; Digits 0-9

    AUTH_CHAR       = ALPHA / DIGIT / "-" / "_"

    auth_type       = 1*20AUTH_CHAR

    auth_command    = "AUTH" SPACE auth_type [SPACE (base64 / "=")]
                      *(CRLF [base64]) CRLF

    base64          = base64_terminal /
                      ( 1*(4base64_CHAR) [base64_terminal] )

    base64_char     = UPALPHA / LOALPHA / DIGIT / "+" / "/"
                      ;; Case-sensitive

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

    continue_req    = "+" SPACE [base64] CRLF

    CR              = %x0C           ;; ASCII CR, carriage return

    CRLF            = CR LF

    LF              = %x0A           ;; ASCII LF, line feed

    SPACE           = %x20           ;; ASCII SP, space
```

4.2.   Examples

Here is an example of a client attempting AUTH PLAIN under TLS and making use of the initial client response:

```
     S: +OK pop.example.com BlurdyBlurp POP3 server ready
     C: CAPA
     S: +OK List of capabilities follows
     S: SASL KERBEROS_V4 GSSAPI ANONYMOUS
     S: STLS
     S: IMPLEMENTATION BlurdyBlurp POP3 server
     S: .
     C: STLS
     S: +OK Begin TLS negotiation now
       ... TLS negotiation proceeds, further commands protected by TLS
layer ...
     C: CAPA
     S: +OK List of capabilities follows
     S: SASL PLAIN KERBEROS_V4 GSSAPI ANONYMOUS
     S: IMPLEMENTATION BlurdyBlurp POP3 server
     S: .
     C: AUTH PLAIN dGVzdAB0ZXN0AHRlc3Q=
     S: +OK Maildrop locked and ready
```

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: CAPA
     S: +OK List of capabilities follows
     S: SASL PLAIN KERBEROS_V4 GSSAPI ANONYMOUS
     S: IMPLEMENTATION BlurdyBlurp POP3 server
     S: .
     C: AUTH PLAIN
       (note that there is a space following the '+' on the following line)
     S: +
     C: dGVzdAB0ZXN0AHRlc3Q=
     S: +OK Maildrop locked and ready
```

Here is an example using a mechanism which does not support an initial client send, and includes server challenges:

```
     S: +OK pop.example.com BlurdyBlurp POP3 server ready
     C: CAPA
     S: +OK List of capabilities follows
     S: SASL KERBEROS_V4 GSSAPI ANONYMOUS
     S: STLS
     S: IMPLEMENTATION BlurdyBlurp POP3 server
     S: .
     C: AUTH KERBEROS_V4
     S: + ezLUFA==
        (the following lines are broken for editorial clarity only)
     C: BAYFQU5EUkVXLkNNVS5FRFUAOCCXeMiVyFe9K6Nwne7+sPLgIoF9YQ5ePfxUsMlJAf
        C7aoNySU8nrqS9m8JAddsUeuyc5HFXXovaKLrZNo2bTLH0Lyolwy0W9ryJDojbKmHy
        zSMqFsGD4EL0
     S: + Z74fTwDw7KQ=
     C: vSAF7ha6qotK2UHUgKlsEA==
     S: +OK Maildrop locked and ready
        ... at this point a security layer has been established and additional
            commands and responses proceed within it ...
```

5. Extended POP3 Response Codes

This section defines four POP3 response codes which can be used to determine the reason for a failed login (provided that the server advertises the RESP-CODES capability [POP3-EXT]). These definitions supercede those in [POP3-EXT] and [POP3-CODES].

It is RECOMMENDED that server applications use these codes when possible to allow clients a straightforward, interoperable way to determine the cause of an authentication failure (as opposed to parsing error text).

5.1. The LOGIN-DELAY Response Code

This occurs on an -ERR response to an AUTH, USER (see note), PASS or APOP command and indicates that the user has logged in recently and will not be allowed to login again until the login delay period has expired.

Please see the Security Considerations section of this document for an important note about returning this code in response to the USER command.

5.2. The IN-USE Response Code

This occurs on an -ERR response to an AUTH, APOP, or PASS command. It indicates the authentication was successful, but the user's maildrop is currently in use (probably by another POP3 client).

5.3. The AUTH Response Code

The AUTH response code informs the client that there is a problem with the user's credentials. This might be an incorrect password, an unknown user name, an expired account, an attempt to authenticate in violation of policy (such as from an invalid location or during an unauthorized time), or some other problem.

The AUTH response code is valid with an -ERR response to any authentication command including AUTH, USER (see the note in the Security Considerations section of this document), PASS, or APOP.

Servers which include the AUTH response code with any authentication failure SHOULD support the CAPA command [POP3-EXT] and SHOULD include the AUTH-RESP-CODE capability (defined in the next section) in the CAPA response. AUTH-RESP-CODE assures the client that only errors with the AUTH code are caused by credential problems.

5.3.1. The AUTH-RESP-CODE Capability

CAPA tag:
    AUTH-RESP-CODE

Arguments:
    none

Added commands:
    none

Standard commands affected:
    none

Announced states / possible differences:
    both / no

Commands valid in states:
    n/a

Specification reference:
    this document

Discussion:
    The AUTH-RESP-CODE capability indicates that the server includes the AUTH response code with any authentication error caused by a problem with the user's credentials.

5.4. The SYS Response Code

The SYS response code announces that a failure is due to a system error, as opposed to the user's credentials or an external condition. It is hierarchical, with two possible second-level codes: TEMP and PERM. (Case is not significant at any level of the hierarchy.)

SYS/TEMP indicates a problem which is likely to be temporary in nature, and therefore there is no need to alarm the user, unless the failure persists. Examples might include a central resource which is currently locked or otherwise temporarily unavailable, insufficient free disk or memory, etc.

SYS/PERM is used for problems which are unlikely to be resolved without intervention. It is appropriate to alert the user and suggest that the organization's support or assistance personnel be contacted. Examples include corrupted mailboxes, system configuration errors, etc.

The SYS response code is valid with an -ERR response to any command.

6. Security Considerations

Security issues are discussed throughout this memo.

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.

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 atleast 3 attempts to authenticate have failed.

Implementations MUST support a configuration where [SASL] mechanisms that are vulnerable to passive eavesdropping attacks (such as [PLAIN]) are not advertised or used without the presence of an external security layer such as [TLS].

Returning the LOGIN-DELAY or AUTH response codes to the USER command avoids the work of authenticating the user but is likely to reveal information to the client about the existence of the account in question. Unless the server is operating in an environment where user names are not secret (for example, many popular email clients advertise the POP server and user name in an outgoing mail header),

or where server access is restricted, or the server can verify that the connection is to the same user, the the server SHOULD NOT issue this response code to the USER command. The server still saves the cost of opening the maildrop, which in some environments is the most expensive step.

7. IANA Considerations

This document requests that the IANA update the entry for the "pop" SASL protocol name to point at this document.

This document requests that the IANA update the entry for the SASL POP3 capability to be as defined in Section 3 of this document.

This document requests that the IANA update the entry for the LOGIN-DELAY, IN-USE, AUTH, SYS/TEMP, and SYS/PERM POP3 response codes to this document.

This document requests that the IANA update the entry for the AUTH-RESP-CODE capability to be as defined in Section 5.3.1 of this document.

8. Protocol Actions

[RFC Editor: Remove this section before publication]

This document obsoletes RFC 1734 and replaces it as a Proposed Standard. By moving RFC 1734 to Historic, RFC 1731 can also be moved to Historic (as RFC 1734 was the last document to have a normative reference).

This document obsoletes RFC 3206 and replaces it as a Proposed Standard.

It also updates information contained in Section 6.3 and Section 8 of RFC 2449.

9. Intellectual Property Rights

The IETF takes no position regarding the validity or scope of any intellectual property 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; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances

10. Copyright

11. References

The following documents contain normative definitions or specifications that are necessary for correct understanding of this protocol:

[BASE64]   Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 3548, July 2003.

[DIGEST-MD5] Leach, P., Melnikov, A., and Newman C., "Using Digest Authentication as a SASL Mechanism", draft-ietf-sasl-rfc2831bis-*.txt, a work in progress.

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

[POP3]   Myers, J. and Rose, M., "Post Office Protocol - Version 3", STD 53, RFC 1939, May 1996.

[POP3-EXT]   Gellens, R., Newman, C., and Lundblade, L., "POP3 Extension Mechanism", RFC 2449, November 1998.

[POP3-TLS]   Newman, C., "Using TLS with IMAP, POP3, and ACAP", RFC 2595, June 1999.

[SASL]   Melnikov, A., "Simple Authentication and Security Layer (SASL)", draft-ietf-sasl-rfc2222bis-*.txt, a work in progress.

[SASLprep]   Zeilega, K., "SASLprep: Stringprep profile for user names and passwords", draft-ietf-sasl-saslprep-*.txt, a work in progress

[StringPrep] Hoffman, P. and Blanchet, M., "Preparation of Internationalized Strings ("stringprep")", draft-hoffman-rfc3454bis-*.txt, a work in progress

The following references are for informational purposes only:

[PLAIN]   Zeilenga, K., "The Plain SASL Mechanism", draft-ietf-sasl-plain-*.txt, a work in progress.

[POP3-AUTH] Myers, J., "POP3 AUTHentication Command", RFC 1734, January 1994.

[POP3-CODES]

Gellens, R., "The SYS and AUTH POP Response Codes", RFC 3206, February 2002.

[TLS] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

12. Changes From RFC 1734, RFC 2449, and RFC 3206

1. The SASL-based semantics defined in RFC 2449 are now normative for the AUTH extension.

2. Clarifications and examples of the proper behavior of initial client response handling.

3. Minimum requirement of support for DIGEST-MD5.

4. Clarify ordering of TLS and SASL security layers.

5. Update references to newer versions of various specifications.

6. Clarify that the mechanism list can change.

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

8. General other editorial clarifications.

9. Consolidation of all applicable information into a single document.

13. Author's Address:

Robert Siemborski
Carnegie Mellon, Andrew Systems Group
Cyert Hall 207
5000 Forbes Avenue
Pittsburgh, PA 15213
+1 412 268 7456
rjs3+@andrew.cmu.edu

14. Acknowledgments:

The author would like to acknowledge the contributions of John Myers, Randall Gellens, Chris Newman, Laurence Lundblade, and other contributors to RFC 1734, RFC 2554, and RFC 3206, on which this document draws heavily.

The author would also like to thank Ken Murchison, Alexey Melnikov, and Mark Crispin for the time they devoted to reviewing early drafts of this document.