Network Working Group   J. G. Myers
Internet Draft: IMAP4 Authentication Mechanisms   Carnegie Mellon
Document: internet-drafts/draft-ietf-imap-auth-01.txt   June 1994

IMAP4 Authentication mechanisms

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. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a ``working draft'' or ``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 ds.internic.net, nic.nordu.net, ftp.isi.edu, or munnari.oz.au.

This is a draft document of the IETF IMAP Working Group. It is a preliminary specification of several authentication mechanisms for use by the AUTHENTICATE command of the IMAP4 protocol.

A revised version of this draft document will be submitted to the RFC editor as a Proposed Standard for the Internet Community. Discussion and suggestions for improvement are requested. This document will expire before 31 December 1994. Distribution of this draft is unlimited. Comments are solicited and should be sent to imap@CAC.Washington.EDU.

Introduction

The Internet Message Access Protocol, Version 4 [IMAP4] contains the AUTHENTICATE command, for identifying and authenticating a user to an IMAP4 server and for optionally negotiating a protection mechanism for subsequent protocol interactions. This document describes several authentication mechanisms for use by the IMAP4 AUTHENTICATE command.

Kerberos version 4 authentication mechanism

The authentication type associated with Kerberos version 4 is ``KERBEROS_V4''.

The data encoded in the first ready response contains a random 32-bit number in network byte order. The client should respond with a Kerberos ticket and an authenticator for the principal "imap.hostname@realm", where "hostname" is the first component of the host name of the server with all letters in lower case and where "realm" is the Kerberos realm of the server. The encrypted checksum field included within the Kerberos authenticator should contain the server provided 32-bit number in network byte order.

Upon decrypting and verifying the ticket and authenticator, the server should verify that the contained checksum field equals the original server provided random 32-bit number. Should the verification be successful, the server must add one to the checksum and construct 8 octets of data, with the first four octets containing the incremented checksum in network byte order, the fifth octet containing a bit-mask specifying the protection mechanisms supported by the server, and the sixth through eighth octets containing, in network byte order, the maximum cipher-text buffer size the server is able to receive. The server must encrypt the 8 octets of data in the session key and issue that encrypted data in a second ready response. The client should consider the server authenticated if the first four octets the un-encrypted data is equal to one plus the checksum it previously sent.

The client must construct data with the first four octets containing the original server-issued checksum in network byte order, the fifth octet containing the bit-mask specifying the selected protection mechanism, the sixth through eighth octets containing in network byte order the maximum cipher-text buffer size the client is able to receive, and the following octets containing a user name string. The client must then append from one to eight octets so that the length of the data is a multiple of eight octets. The client must then PCBC encrypt the data with the session key and respond to the second ready response with the encrypted data. The server decrypts the data and verifies the contained checksum. The username field identifies the user for whom subsequent IMAP operations are to be performed; the server must verify that the principal identified in the Kerberos ticket is authorized to connect as that user. After these verifications, the authentication process is complete.

The protection mechanisms and their corresponding bit-masks are as follows:

```
1 No protection mechanism
2 Integrity (krb_mk_safe) protection
4 Privacy (krb_mk_priv) protection
```

EXAMPLE: The following are two Kerberos version 4 login scenarios (note that the line breaks in the sample authenticators are for editorial clarity and are not in real authenticators)

```
S: * OK IMAP4 Server
C: A001 AUTHENTICATE KERBEROS_V4
S: + AmFYig==
C: BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3/IJmrMG+25a4DT
   +nZImJjnTNHJUtxAA+o0KPKfHEcAFs9a3CL5Oebe/ydHJUwYFd
   WwuQ1MWiy6IesKvjL5rL9WjXUb9MwT9bpObYLGOKi1Qh
S: + or//EoAADZI=
C: DiAF5A4gA+oOIALuBkAAmw==
S: A001 OK Kerberos V4 authentication successful


S: * OK IMAP4 Server
C: A001 AUTHENTICATE KERBEROS_V4
S: + gcfgCA==
C: BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3/IJmrMG+25a4DT
   +nZImJjnTNHJUtxAA+o0KPKfHEcAFs9a3CL5Oebe/ydHJUwYFd
   WwuQ1MWiy6IesKvjL5rL9WjXUb9MwT9bpObYLGOKi1Qh
S: A001 NO Kerberos V4 authentication failed
```

GSSAPI authentication mechanism

The authentication type associated with all mechanisms employing the GSSAPI [RFC1508] is ``GSSAPI''.

The first ready response issued by the server contains no data. The client should call GSS_Init_sec_context, passing in 0 for input_context_handle (initially) and a targ_name equal to output_name from GSS_Import_Name called with input_name_type of NULL and input_name_string of "SERVICE:imap@hostname" where "hostname" is the fully qualified host name of the server with all letters in lower case. The client must then respond with the resulting output_token. If GSS_Init_sec_context returns GSS_CONTINUE_NEEDED, then the client should expect the server to issue a token in a subsequent ready response. The client must pass the token to another call to GSS_Init_sec_context.

If GSS_Init_sec_context returns GSS_COMPLETE, then the client should

respond with any resulting output_token. If there is no output_token, the client should respond with no data. The client should then expect the server to issue a token in a subsequent ready response. The client should pass this token to GSS_Unseal and interpret the first octet of resulting cleartext as a bit-mask specifying the protection mechanisms supported by the server and the second through fourth octets as the maximum size output_message to send to the server. The client should construct data, with the first octet containing the bit-mask specifying the selected protection mechanism, the second through fourth octets containing in network byte order the maximum size output_message the client is able to receive, and the remaining octets containing a user name string. The client must pass the data to GSS_Seal with conf_flag set to FALSE, and respond with the generated output_message. The client can then consider the server authenticated.

The server must issue a ready response with no data and pass the resulting client supplied token to GSS_Accept_sec_context as input_token, setting acceptor_cred_handle to NULL (for "use default credentials"), and 0 for input_context_handle (initially). If GSS_Accept_sec_context returns GSS_CONTINUE_NEEDED, the server should return the generated output_token to the client in a ready response and pass the resulting client supplied token to another call to GSS_Accept_sec_context.

If GSS_Accept_sec_context returns GSS_COMPLETE, then if an output_token is returned, the server should return it to the client in a ready response and expect a reply from the client with no data. Whether or not an output_token was returned, the server then should then construct 4 octets of data, with the first octet containing a bit-mask specifying the protection mechanisms supported by the server and the second through fourth octets containing in network byte order the maximum size output_token the server is able to receive. The server must then pass the plaintext to GSS_Seal with conf_flag set to FALSE and issue the generated output_message to the client in a ready response. The server must then pass the resulting client supplied token to GSS_Unseal and interpret the first octet of resulting cleartext as the bit-mask for the selected protection mechanism, the second through fourth octets as the maximum size output_message to send to the client, and the remaining octets as the user name. Upon verifying the src_name is authorized to authenticate as the user name, The server should then consider the client authenticated.

The protection mechanisms and their corresponding bit-masks are as follows:

1 No protection mechanism
2 Integrity protection.

        Sender calls GSS_Seal with conf_flag set to FALSE
      4 Privacy protection.
        Sender calls GSS_Seal with conf_flag set to TRUE

S/Key authentication mechanism

The authentication type associated with S/Key [SKEY] is ``SKEY''.

The first ready response issued by the server contains no data. The client responds with the user name string.

The data encoded in the second ready response contains the decimal sequence number followed by a single space and the seed string for the indicated user. The client responds with the one-time-password, as either a 64-bit value in network byte order or encoded in the "six English words" format.

Upon successful verification of the one-time-password, the server should consider the client authenticated.

S/Key authentication does not provide for any protection mechanisms.

EXAMPLE: The following are two S/Key login scenarios.

```
S: * OK IMAP4 Server
C: A001 AUTHENTICATE SKEY
S: +
C: bW9yZ2Fu
S: + OTUgUWE1ODMwOA==
C: Rk9VUiBNQU5OIFNPT04gRklSIFZBUlkgTUFTSA==
S: A001 OK S/Key authentication successful


S: * OK IMAP4 Server
C: A001 AUTHENTICATE SKEY
S: +
C: c21pdGg=
S: + OTUgUWE1ODMwOA==
C: BsAY3g4gBNo=
S: A001 NO S/Key authentication failed
```

References

[IMAP4] Crispin, Mark R., "Internet Message Access Protocol - Version 4", Work in Progress of the IETF IMAP WG, draft-ietf-imap-imap4-??.txt. Check Internet Drafts listing for latest version.

[RFC1508] Linn, John, "Generic Security Service Application Program Interface", RFC 1508.

[SKEY] Haller, Neil M. "The S/Key One-Time Password System", Bellcore, Morristown, New Jersey, October 1993, thumper.bellcore.com:pub/nmh/docs/ISOC.symp.ps

Security Considerations

Security issues are discussed throughout this memo.

Author's Address

John G. Myers
Carnegie-Mellon University
5000 Forbes Ave.
Pittsburgh PA, 15213-3890

Email: jgm+@cmu.edu

This document will expire before December 25, 1994.