NAME

gss_unwrap, **gss_unseal -** Convert a message previously protected by gss_wrap(3) back to a usable form

SYNOPSIS

#include <gssapi/gssapi.h>

OM_uint32
gss_unwrap(*OM_uint32 *minor_status*, *const gss_ctx_id_t context_handle*,
  const gss_buffer_t input_message_buffer, *gss_buffer_t output_message_buffer*, *int *conf_state*,
  *gss_qop_t *qop_state*);

OM_uint32
gss_unseal(*OM_uint32 *minor_status*, *gss_ctx_id_t context_handle*,
  gss_buffer_t input_message_buffer, *gss_buffer_t output_message_buffer*, *int *conf_state*,
  *gss_qop_t *qop_state*);

DESCRIPTION

Converts a message previously protected by gss_wrap(3) back to a usable form, verifying the embedded MIC. The conf_state parameter indicates whether the message was encrypted; the qop_state parameter indicates the strength of protection that was used to provide the confidentiality and integrity services.

Since some application-level protocols may wish to use tokens emitted by gss_wrap(3) to provide "secure framing", implementations must support the wrapping and unwrapping of zero-length messages.

The **gss_unseal**() routine is an obsolete variant of **gss_unwrap**(). It is provided for backwards compatibility with applications using the GSS-API V1 interface. A distinct entrypoint (as opposed to #define) is provided, both to allow GSS-API V1 applications to link and to retain the slight parameter type differences between the obsolete versions of this routine and its current form.

PARAMETERS

minor_status Mechanism specific status code.

context_handle Identifies the context on which the message arrived.

input_message_buffer Protected message.

output_message_buffer Buffer to receive unwrapped message. Storage associated with this buffer must be freed by the application after use with a call to gss_release_buffer(3).

conf_state

Non-zero Confidentiality and integrity protection were used.

Zero Integrity service only was used.

Specify NULL if not required.

qop_state Quality of protection provided. Specify NULL if not required.

RETURN VALUES

GSS_S_COMPLETE Successful completion.

GSS_S_DEFECTIVE_TOKEN The token failed consistency checks.

GSS_S_BAD_SIG The MIC was incorrect

GSS_S_DUPLICATE_TOKEN The token was valid, and contained a correct MIC for the message, but it had already been processed.

GSS_S_OLD_TOKEN The token was valid, and contained a correct MIC for the message, but it is too old to check for duplication.

GSS_S_UNSEQ_TOKEN The token was valid, and contained a correct MIC for the message, but has been verified out of sequence; a later token has already been received.

GSS_S_GAP_TOKEN The token was valid, and contained a correct MIC for the message, but has been verified out of sequence; an earlier expected token has not yet been received.

GSS_S_CONTEXT_EXPIRED The context has already expired.

GSS_S_NO_CONTEXT The context_handle parameter did not identify a valid context.

SEE ALSO

gss_release_buffer(3), gss_wrap(3)

STANDARDS

RFC 2743 Generic Security Service Application Program Interface Version 2, Update 1

RFC 2744 Generic Security Service API Version 2 : C-bindings

HISTORY

The **gss_unwrap** function first appeared in FreeBSD 7.0.

AUTHORS

John Wray, Iris Associates

COPYRIGHT