Internet Draft
Common Authentication Technology Working Group
draft-ietf-cat-gss-conv-00.txt

T. Ts'o
MIT

October 21, 1999

GSS Conversation C-bindings Interface

Status of this Memo

1. Abstract

Traditionally, the GSSAPI specification has not included in its scope the acquisition of initial credentials, or mechanisms which have required interaction with the user in the course of the security context. This has limited the applicability of the GSSAPI specification.

This specification allows an application program to register a callback function so that a GSSAPI mechanism can request conversation services from the application program. The application program is responsible for displaying messages to the user and requesting input from the user via the callback function.

The goal of this specification is to allow certain low-

infrastructure-requiring mechanisms to prompt the user for a username, password, SecureID response, etc. It might also be used by mechanisms such as Kerberos when the GSSAPI library wishes to obtain initial credentials itself, instead of assuming that they are already present in the process.

2. GSSAPI Conversation API

2.1 The gss_acquire_cred_conv function

```
OM_uint32 gss_acquire_cred_conv (
        OM_uint32         *minor_status,
        const gss_name_t  desired_name,
        OM_uint32         time_req,
        const gss_OID_set desired_mechs,
        gss_cred_usage_t  cred_usage,
        gss_cred_conv_t   conversation_info,
        gss_cred_id_t     *output_cred_handle,
        gss_OID_set       *actual_mechs,
        OM_uint32         *time_rec)
```

This function is an extended version of the gss_acquire_cred function defined in the GSSAPI C-bindings specification, with the addition of a new parameter, conversation_info. If the conversation_info parameter is NULL, then the behavior of gss_acquire_cred_conv will be identical to gss_acquire_cred.

If a conversation function is registered via the conversation_info parameter, then the GSSAPI mechanism may call the conversation function during the execution of the gss_acquire_cred_conv() function, and the returned GSSAPI credentials will have a reference to the conversation function. If GSSAPI credential containing a conversation function is passed to gss_init_sec_context(), gss_accept_sec_context(), or gss_add_cred(), the GSSAPI mechanism may also call the conversation function during the course of execution of those GSSAPI functions.

2.2 The gss_copy_buffer function

```
OM_uint32 gss_display_name (
        OM_uint32       *minor_status,
        gss_buffer_t    input_buffer,
        gss_buffer_t    output_buffer)
```

This function is used to copy a buffer into dynamically allocated memory which can be freed by the gss_release_buffer() call.

The conversation function must use this function when filling in buffers in the gss_resp array, since the GSSAPI library will free the buffer

after it is done using it. (Otherwise, there is no good way to make sure the buffers containing the user's responses get freed.)

3 Data Types and Structures

3.1 The gss_cred_conv_t data structure

The gss_cred_conv_t data type contains a pointer to the conversation function and private data pointer which is passed to the conversation function. It shall have the following C definition:

```
typedef struct gss_conv {
        int (*conv)(OM_uint32 gss_num_msg,
                    const gss_conv_message_t *gss_msg,
                    gss_conv_message_t *gss_resp,
                    void *gss_appdata_ptr);
        void *gss_appdata_ptr;
} *gss_cred_conv_t;
```

The first parameter to the conversation function, gss_num_msg, specifies the number of messages to be processed by the conversation function. The second and third parameters are two arrays which have gss_num_msg entries. The first array, gss_msg, contains a list of text messages which can be prompts or informational text to be displayed to the user. The second array, gss_resp, is to be filled in by the conversation function as defined below. The final parameter is a private data pointer which was supplied by the calling application in the gss_cred_conv_t structure.

The conversation function returns the following integer codes to its caller (the GSSAPI library):

```
#define GSS_CONV_SUCCESS        0
        /* The user filled in the requested fields and all is well
*/

#define GSS_CONV_ABORT          1
        /* The user requested an abort */
```

3.2 The gss_conv_message_t structure

The gss_conv_message_t structure is the data element which is used to pass prompts and informational messages to the conversation function, as well as passing the user's responses back to the GSSAPI library. It has the following C type definition:

```
typedef struct gss_conv_message {
        OMuint32                gss_msg_code;
```

```
                gss_buffer_desc         gss_msg;
        } gss_conv_message_t;
```

In the gss_msg array, the gss_msg_code field specifies whether each item in the gss_msg array is a prompt for user response, or informative text to be displayed to the user. If the entry in the gss_msg array is a prompt, the gss_msg_code further specifies whether the user response should be echoed or not. If the entry in the gss_msg array is an informative text, the gss_msg_code also specifies whether it should be displayed as an error or as normal informative text.

The following numbers are defined for the gss_msg_code field:

```
        #define GSS_CONV_ECHO_OFF       1
                /* Obtain a string without echoing any text */

        #define GSS_CONV_ECHO_ON        2
                /* Obtain a string with echoing enabled */

        #define GSS_CONV_ERROR_MSG      3
                /* Display an error */

        #define GSS_CONV_TEXT_INFO      4
                /* Display some informational text. */
```

In the gss_resp array, the gss_msg_code field is reserved, and should be set to zero by the conversation function.

4. Operation of the conversation function

If a conversation function has been registered with a GSSAPI credential, it may be called during the original call to gss_acquire_cred_conv, or when one of the following functions is called with the GSSAPI credential with the conversation function: gss_init_sec_context(), gss_accept_sec_context(), or gss_acquire_cred().

The conversation function supplied by the application shall process each entry in the gss_msg array in order. If the entry in the gss_msg array is an informational or error text, the conversation function must arrange to have the string displayed to the user. If the entry in the gss_msg array is a prompt, the conversation function is responsible for displaying the prompt and requesting input from the user, with echoing enabled or disabled as appropriate.

The response from the user should be placed in the corresponding entry in the gss_resp array, in a dynamically allocated gss_buffer_t. The conversation function should not modify entries in the gss_resp array which correspond to informative texts or error messages in the gss_msg

array. The GSSAPI library is responsible for freeing the dynamically allocated buffers placed in the gss_resp array by the conversation function. These buffers should be freed before the GSSAPI function which called the conversation function returns to the application program.

5. Security Considerations

This interfaces allows a GSSAPI library to ask the user for passwords and other authentication information. It is the responsibility of the calling application to zero any memory that might contain user responses to prevent them from being exposed in case of a program crash.

6. Author's Address

Theodore Ts'o
VA Linux Systems
43 Pleasant St.
Medford, MA 02155

Phone: (781) 391-3464

EMail: tytso@valinux.com