

NETWORK WORKING GROUP
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2009

N. Williams
Sun
March 9, 2009

**GSS-API Extension for Storing Delegated Credentials
draft-ietf-kitten-gssapi-store-cred-04.txt**

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

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 and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on September 10, 2009.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (<http://trustee.ietf.org/license-info>). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

This document defines a new function for the GSS-API which allows applications to store delegated (and other) credentials in the implicit GSS-API credential store. This is needed for GSS-API applications to use delegated credentials as they would use other credentials.

Table of Contents

1.	Conventions used in this document	3
2.	Introduction	4
3.	GSS_Store_cred()	5
4.	C-Bindings	7
5.	Examples	8
6.	IANA considerations	9
7.	Security considerations	10
8.	Normative References	11
	Author's Address	12

1. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119](#)].

2. Introduction

The GSS-API [[RFC2743](#)] clearly assumes that credentials exist in an implicit store whence they can be acquired using `GSS_Acquire_cred()` and `GSS_Add_cred()` or through use of the default credential. Multiple credential stores may exist on a given host, but only one store may be accessed by `GSS_Acquire_cred()` and `GSS_Add_cred()` at any given time.

This assumption can be seen in sections [1.1.1.2](#) and [1.1.1.3](#) of [[RFC2743](#)] as well as in [section 3.5 of \[RFC2744\]](#).

Applications may be able to change the credential store from which credentials can be acquired, either by changing user contexts (where the applications have the privilege to do so) or by other means (where a user may have multiple credential stores).

Some GSS-API acceptor applications always change user contexts, after accepting a GSS-API security context and making appropriate authorization checks, to the user context corresponding to the initiator principal name or to a context requested by the initiator. The means by which credential stores are managed are generally beyond the scope of the GSS-API.

In the case of delegated credential handles however, such credentials do not exist in the acceptor's credential store or in the credential stores of the user contexts to which the acceptor application might change. The GSS-API provides no mechanism by which delegated credential handles can be made available for acquisition through `GSS_Acquire_cred()/GSS_Add_cred()`. The GSS-API also does not provide any credential import/export interfaces like the GSS-API context import/export interfaces.

Thus acceptors are limited to making only direct use of delegated credential handles and only with `GSS_Init_sec_context()`, `GSS_Inquire_cred*()` and `GSS_Release_cred()`. This limitation is particularly onerous on Unix systems where a call to `exec()` to replace the process image obliterates any delegated credentials handle that may exist in that process.

In order to make delegated credentials generally as useful as credentials that can be acquired with `GSS_Acquire_cred()` and `GSS_Add_cred()` a primitive is needed which allows storing of credentials in the implicit credential store. This primitive we call "`GSS_Store_cred()`."

3. GSS_Store_cred()

Inputs:

- o `input_cred_handle` CREDENTIAL HANDLE, -- credential to store; MUST NOT be `GSS_C_NO_CREDENTIAL`
- o `cred_usage` INTEGER -- 0=INITIATE-AND-ACCEPT, 1=INITIATE-ONLY, 2=ACCEPT-ONLY
- o `desired_mech_element` OBJECT IDENTIFIER, -- if `GSS_C_NULL_OID` then store all the elements of the `input_cred_handle`, otherwise store only the element of the corresponding mechanism
- o `overwrite_cred` BOOLEAN, -- if TRUE replace any credential for the same principal in the credential store
- o `default_cred` BOOLEAN -- if TRUE make the stored credential available as the default credential (for acquisition with `GSS_C_NO_NAME` as the desired name or for use as `GSS_C_NO_CREDENTIAL`)

Outputs:

- o `major_status` INTEGER,
- o `minor_status` INTEGER,
- o `mech_elements_stored` SET OF OBJECT IDENTIFIER, -- the set of mechanism OIDs for which credential elements were successfully stored
- o `cred_usage_stored` INTEGER -- like `cred_usage`, but indicates what kind of credential was stored (useful when the `cred_usage` input parameter is set to `INITIATE-AND-ACCEPT`)

Return `major_status` codes:

- o `GSS_S_COMPLETE` indicates that the credentials were successfully stored.
- o `GSS_S_CREDENTIALS_EXPIRED` indicates that the input credentials had expired or expired before they could be stored.
- o `GSS_S_NO_CRED` indicates that no input credentials were given.
- o `GSS_S_UNAVAILABLE` indicates that the credential store is not available.

- o `GSS_S_DUPLICATE_ELEMENT` indicates that an element of the input credential could not be stored because a credential for the same principal exists in the current credential store and the `overwrite_cred` input argument was `FALSE`.
- o `GSS_S_FAILURE` indicates that the credential could not be stored for some other reason. The minor status code may provide more information if a non-`GSS_C_NULL_OID` `desired_mech_element` was given.

`GSS_Store_cred()` is used to store, in the current credential store, a given credential that has either been acquired from a different credential store or been accepted as a delegated credential.

Specific mechanism elements of a credential can be stored one at a time by specifying a non-`GSS_C_NULL_OID` mechanism OID as the `desired_mech_element` input argument, in which case the minor status output `SHOULD` have a mechanism-specific value when the major status is not `GSS_S_COMPLETE`.

The initiator, acceptor or both usages of the input credential may be stored as per the `cred_usage` input argument.

The credential elements that were actually stored, when the major status is `GSS_S_COMPLETE`, are indicated through the `cred_usage_stored` and `mech_elements_stored` function outputs.

If credentials already exist in the current store for the principal of the `input_cred_handle`, then those credentials are not replaced with the input credentials unless the `overwrite_cred` input argument is `TRUE`.

Finally, if the current credential store has no default credential (that is, no credential that could be acquired for `GSS_C_NO_NAME`) or if the `default_cred` input argument is `TRUE`, and the input credential can be successfully stored, then the input credential will be available for acquisition with `GSS_C_NO_NAME` as the desired name input to `GSS_Acquire_cred()` or `GSS_Add_cred()` as well as for use as `GSS_C_NO_CREDENTIAL` for the `cred_handle` inputs to `GSS_Inquire_cred()`, `GSS_Inquire_cred_by_mech()`, `GSS_Init_sec_context()` and `GSS_Accept_sec_context()`.

4. C-Bindings

The C-bindings for GSS_Store_cred() make use of types from and are designed based on the style of the GSS-APIv2 C-Bindings [[RFC2744](#)].

```
OM_uint32 gss_store_cred(  
    OM_uint32      *minor_status,  
    gss_cred_id_t  input_cred_handle,  
    gss_cred_usage_t cred_usage,  
    const gss_OID   desired_mech,  
    OM_uint32      overwrite_cred,  
    OM_uint32      default_cred,  
    gss_OID_set     *elements_stored,  
    gss_cred_usage_t *cred_usage_stored)
```

Figure 1

The two boolean arguments, 'overwrite_cred' and 'default_cred' are typed as OM_uint32; 0 corresponds to FALSE, non-zero values correspond to TRUE.

5. Examples

The intended usage of `GSS_Store_cred()` is to make delegated credentials available to child processes of GSS-API acceptor applications. Example pseudo-code:

```
/*
 * <GSS_Accept_sec_context() loop resulting in GSS_S_COMPLETE,
 * an initiator name (hereafter, "src_name") and a delegated
 * credential handle (hereafter "deleg_cred").>
 *
 * <"requested_username" is a username derived from the
 * initiator name or explicitly requested by the initiator
 * application.>
 */
...

if (authorize_gss_client(src_name, requested_username)) {
    /*
     * For Unix-type platforms this may mean calling setuid() and
     * it may or may not also mean setting/unsetting such
     * environment variables as KRB5CCNAME and what not -- all
     * OS-specific details.
     */
    if (change_user_context(requested_username))
        (void) gss_store_creds(&minor_status, deleg_cred,
                               GSS_C_INITIATE, actual_mech,
                               0, 1, NULL, NULL);
    }
    else ...
}
else ...
```


6. IANA considerations

There are no IANA considerations in this document.

7. Security considerations

Acceptor applications **MUST** only store delegated credentials into appropriate credential stores and only after proper authorization of the authenticated initiator principal to the requested service(s).

Acceptor applications that have no use for delegated credentials **MUST** release them (such acceptor applications that use the GSS-API C-Bindings may simply provide a NULL value for the `delegated_cred_handle` argument to `gss_accept_sec_context()`).

8. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", [RFC 2743](#), January 2000.
- [RFC2744] Wray, J., "Generic Security Service API Version 2 : C-bindings", [RFC 2744](#), January 2000.

Author's Address

Nicolas Williams
Sun Microsystems
5300 Riata Trace Ct
Austin, TX 78727
US

Email: Nicolas.Williams@sun.com