KITTEN WORKING GROUP	N. Williams
Internet-Draft	Sun
Intended status: Standards Track	L. Johansson
Expires: January 31, 2010	SUNET
	July 30, 2009

GSS-API Naming Extensions
draft-ietf-kitten-gssapi-naming-exts-05.txt

Abstract

The Generic Security Services API (GSS-API) provides a simple naming architecture that supports name-based authorization. This document introduces new APIs that extend the GSS-API naming model to support name attribute transfer between GSS-API peers.

Table of Contents

---

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] (Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," March 1997.) .

---

2. Introduction

As described in [I-D.GSS-NAMING] (Hartman, S., “Desired Enhancements to GSSAPI Naming,” February 2005.) the GSS-API's naming architecture suffers from certain limitations. This document proposes concrete GSS-API extensions as outlined in [I-D.GSS-NAMING] (Hartman, S., “Desired Enhancements to GSSAPI Naming,” February 2005.).
A number of extensions to the GSS-API [RFC2743] (Linn, J., “Generic Security Service Application Program Interface Version 2, Update 1,” January 2000.) and its C Bindings [RFC2744] (Wray, J., “Generic Security Service API Version 2 : C-bindings,” January 2000.) are described herein. The goal is to make information modeled as "name attributes" available to applications. Such information MAY for instance be used by applications to make authorization-decisions. For example, Kerberos V authorization data elements, both, in their raw forms as well as mapped to more useful value types, can be made available to GSS-API applications through these interfaces.
The model is that GSS names have attributes. The attributes of a name may be authenticated (eg an X509 attribute certificate or signed SAML attribute assertion), or may have been set on a GSS name for the purpose of locally "asserting" the attribute during credential acquisition or security context exchange. Name attributes' values are network representations thereof (e.g., the actual value octets of the contents of an X.509 certificate extension, for example) and are intended to be useful for constructing portable access control facilities. Applications may often require language- or platform-specific data types, rather than network representations of name attributes, so a function is provided to obtain objects of such types associated with names and name attributes.

---

3. Name Attribute Authenticity

TOC

An attribute is 'authenticated' iff there is a secure association between the attribute (and its values) and the trusted source of the peer credential. Examples of authenticated attributes are (any part of) the signed portion of an X.509 certificate or AD-KDCIssued authorization-data elements in Kerberos V Tickets provided of course that the authenticity of the respective security associations (eg signatures) have been verified.
Note that the fact that an attribute is authenticated does not imply anything about the semantics of the attribute nor that the trusted credential source authorized any one semantic of the attribute. Such interpretations MAY be the result of applying local policy to the attribute.
An un-authentciated attribute is called *asserted* in what follows.This is not to be confused with other uses of the word asserted or assertion

eg "SAML attribute assertion", the attributes of which may be authenticated in the sense of this document if the SAML attribute assertion was signed by a key trusted by the peer.

4. Name Attributes/Values as ACL Subjects

TOC

Some name attributes (e.g., numeric user or group identifiers) may be useful as subjects of access control list (ACL) entries, some may not (e.g., time of day login restrictions). The GSS_Inquire_name_attribute() function indicates this.
To facilitate the development of portable applications that make use of name attributes to construct and evaluate portable ACLs the GSS-API makes name attribute values available in canonical network encodings thereof.
To facilitate the development of platform- or language-specific applications that need access to native types of representations of name attributes an optional facility is provided, GSS_Map_name_to_any().

5. Attribute Name Syntax

TOC

Attribute names are represented as opaque STRING elements in the API described below. These attribute names have syntax and semantics that are understood by the application and by the lower-layer implementations (some of which are described below). In order to present a consistent namespace to the application and at the same time impose as few transformation requirements as possible to lower-layer implementations attribute names SHOULD be URIs.
Technologies used in lower-layer protocols may of course use attribute naming that are not based on URIs. Notably X.509 certificates will use OIDs for most naming purposes. In this case OIDs MUST be mapped into URIs.
When mapping entities named by OIDs into this API [RFC3001] (Mealling, M., "A URN Namespace of Object Identifiers," November 2000.) MUST be used. For example if the OID 1.2.3 denotes an Extended Key Usage, the corresponding GSS-API attribute MUST be represented as urn:oid:1.2.3.

6. Mapping Mechanism Facilities to Name Attributes

TOC

In this section we describe two important examples of lower-layer implementations of this API. These examples are not mandatory to

implement and are only provided for reference. The use of [RFC2119] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.)-terms in this section is limited to those implementations of the GSS-API naming extensions that choose to implement these lower-layer technologies.
Kerberos V [RFC4120] (Neuman, C., Yu, T., Hartman, S., and K. Raeburn, “The Kerberos Network Authentication Service (V5),” July 2005.) and the Simple Public-Key GSS-API Mechanism, SPKM described in [RFC2025] (Adams, C., “The Simple Public-Key GSS-API Mechanism (SPKM),” October 1996.), both support the concept and encoding of containers of "authorization-data" as described in [RFC4120] (Neuman, C., Yu, T., Hartman, S., and K. Raeburn, “The Kerberos Network Authentication Service (V5),” July 2005.).
PKIX [RFC5280] (Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” May 2008.) supports a number of attribute-like features, like Extended Key Usage values (EKUs) and certificate extensions.

6.1. Kerberos V and SPKM Authorization-Data

Authorization-data non-container elements asserted in Kerberos V AP-REQ Authenticators MUST be mapped into **asserted** GSS-API name attributes. Authorization-data included in Kerberos V Tickets that is not contained in AD-KDCIssued (with valid signature) MUST be mapped into **asserted** GSS-API name attributes. Conversely, authorization-data elements in Kerberos V Tickets contained by AD-KDCIssued MUST be mapped into **authenticated** GSS-API name attributes.
The URIs for authorization-data elements MUST be the authorization-data elements 'ad-type' prefixed by the IANA-allocated URN prefix (<TBD>)

6.2. PKIX

6.2.1. Standard PKIX Certificate Extensions

PKI certificate extensions MAY/SHOULD/MUST (see comment above) be represented as **authenticated** GSS-API name attributes named using the *same*.
SubjectAltNames and EKUs, specifically, MUST be represented as **authenticated** GSS-API name attributes; see below. Certificate

extensions MUST be represented as GSS-API name attributes named using the OIDs used for the extensions (represented as URNs)
Extended Key Usage extensions, specifically, MUST be mapped as described above, except that GSS-API name attributes for EKUs MUST have NULL values (i.e., zero-length OCTET STRINGs).
PKI certificate key usages (KUs, but not EKUs), MUST NOT be represented as GSS-API name attributes.
PKI certificate subjectAltNames MUST be mapped as **authenticated**.

6.2.2. Other PKIX Certificate Extensions and Attributes

TOC

Any X.509 certificate extension not covered above SHOULD be represented as GSS-AOI name attributes with the OID of the X.509 extension and with OCTET STRING values containing the encoded value of the extension.

6.3. SAML attribute assertions

TOC

Attributes contained in SAML attribute assertions are mapped to GSS-API name attributes with the same URIs as used in the SAML attribute names (subject to representing OIDs to URIs).
SAML attributes found in SAML attribute assertions MUST NOT be mapped as authenticated unless the SAML attribute assertion was signed by a key trusted by the peer or otherwise protected from unauthorized modification.

---

7. API

TOC

7.1. GSS_Display_name_ext()

TOC

Inputs:

*name NAME,

*display_as_name_type OBJECT IDENTIFIER

Outputs:

- *major_status INTEGER,
- *minor_status INTEGER,
- *display_name STRING

Return major_status codes:

- *GSS_S_COMPLETE indicates no error.
- *GSS_S_UNAVAILABLE indicates that the given name could not be displayed using the syntax of the given name type.
- *GSS_S_FAILURE indicates a general error.

This function displays a given name using the given name syntax, if possible. This operation may require mapping MNs to generic name syntaxes or generic name syntaxes to mechanism-specific name syntaxes; such mappings may not always be feasible and MAY be inexact or lossy, therefore this function may fail.

7.1.1. C-Bindings

TOC

```
OM_uint32 GSS_Display_name_ext(
  OM_uint32                      *minor_status,
  gss_name_t                     name,
  gss_OID                        display_as_name_type,
  gss_buffer_t                   display_name
);
```

7.2. GSS_Inquire_name()

TOC

Inputs:

- *name NAME

Outputs:

- *major_status INTEGER,
- *minor_status INTEGER,
- *name_is_MN BOOLEAN,
- *mn_mech OBJECT IDENTIFIER,
- *asserted_attrs SET OF STRING,
- *authenticated_attrs SET OF STRING,
- *all_attrs SET OF STRING,

Return major_status codes:

- *GSS_S_COMPLETE indicates no error.
- *GSS_S_FAILURE indicates a general error.

This function outputs the sets (represented as a NULL terminated array of gss_buffer_t) of attributes of a name, that are authenticated or asserted. It also indicates if a given NAME is an MN or not and, if it is, what mechanism it's an MN of.

7.2.1. C-Bindings

TOC

```
OM_uint32 gss_inquire_name(
  OM_uint32                     *minor_status,
  gss_name_t                    name,
  int                           name_is_MN,
  gss_OID                       *MN_mech,
  gss_buffer_t                  *authenticated,
  gss_buffer_t                  *asserted,
  gss_buffer_t                  *all_attrs
);
```

TOC

7.3. GSS_Get_name_attribute()

Inputs:

- *name NAME,
- *attr STRING

Outputs:

- *major_status INTEGER,
- *minor_status INTEGER,
- *authenticated BOOLEAN, -- TRUE iff authenticated by the trusted peer credential source.
- *complete BOOLEAN -- TRUE iff this represents a complete set of values for the name.
- *values SET OF OCTET STRING,
- *display_values SET OF STRING

Return major_status codes:

- *GSS_S_COMPLETE indicates no error.
- *GSS_S_UNAVAILABLE indicates that the given attribute OID is not known or set.
- *GSS_S_FAILURE indicates a general error.

This function outputs the value(s) associated with a given GSS name object for a given name attribute.
The complete flag denotes that (if TRUE) the set of values represents a complete set of values for this name. The peer being an authoritative source of information for this attribute is a sufficient condition for the complete flag to be set by the peer.
In the federated case when several peers may hold some of the attributes about a name this flag may be highly dangerous and SHOULD NOT be used.
NOTE: This function relies on the GSS-API notion of "SET OF" allowing for order preservation; this has been discussed on the KITTEN WG mailing list and the consensus seems to be that, indeed, that was always the intention. It should be noted however that the order presented does not always reflect an underlying order of the mechanism specific source of the attribute values.

7.3.1. C-Bindings

TOC

The C-bindings of GSS_Get_name_attribute() requires one function call per-attribute value, for multi-valued name attributes. This is done by using a single gss_buffer_t for each value and an input/output integer parameter to distinguish initial and subsequent calls and to indicate when all values have been obtained.
The 'more' input/output parameter should point to an integer variable whose value, on first call to gss_name_attribute_get() MUST be -1, and whose value upon function call return will be non-zero to indicate that additional values remain, or zero to indicate that no values remain. The caller should not modify this parameter after the initial call.

```
     OM_uint32 gss_get_name_attribute(
       OM_uint32                      *minor_status,
       gss_name_t                     name,
       gss_buffer_t                   attr,
       int                            *authenticated,
       int                            *complete,
       gss_buffer_t                   value,
       gss_buffer_t                   display_value,
       int                            *more
     );
```

7.4. GSS_Set_name_attribute()

TOC

Inputs:

```
     *name NAME,

     *complete BOOLEAN, -- TRUE iff this represents a complete set of
      values for the name.

     *attr STRING,

     *values SET OF OCTET STRING
```

Outputs:

```
     *major_status INTEGER,
```

*minor_status INTEGER

Return major_status codes:

- *GSS_S_COMPLETE indicates no error.

- *GSS_S_UNAVAILABLE indicates that the given attribute OID is not known or could not be set.

- *GSS_S_FAILURE indicates a general error.

The complete flag denotes that (if TRUE) the set of values represents a complete set of values for this name. The peer being an authoritative source of information for this attribute is a sufficient condition for the complete flag to be set by the peer.
In the federated case when several peers may hold some of the attributes about a name this flag may be highly dangerous and SHOULD NOT be used.
NOTE: This function relies on the GSS-API notion of "SET OF" allowing for order preservation; this has been discussed on the KITTEN WG mailing list and the consensus seems to be that, indeed, that was always the intention. It should be noted that underlying mechanisms may not respect the given order.

7.4.1. C-Bindings

The C-bindings of GSS_Set_name_attribute() requires one function call per-attribute value, for multi-valued name attributes -- each call adds one value. To replace an attribute's every value delete the attribute's values first with GSS_Delete_name_attribute().

```
OM_uint32 gss_set_name_attribute(
  OM_uint32                     *minor_status,
  gss_name_t                    name,
  int                           complete,
  gss_buffer_t                  attr,
  gss_buffer_t                  value
);
```

7.5. GSS_Delete_name_attribute()

Inputs:

- *name NAME,
- *attr STRING,

Outputs:

- *major_status INTEGER,
- *minor_status INTEGER

Return major_status codes:

- *GSS_S_COMPLETE indicates no error.
- *GSS_S_UNAVAILABLE indicates that the given attribute OID is not known.
- *GSS_S_UNAUTHORIZED indicates that a forbidden delete operation was attempted eg deleting a negative attribute.
- *GSS_S_FAILURE indicates a general error.

Deletion of negative authenticated attributes from NAME objects MUST NOT be allowed and must result in a GSS_S_UNAUTHORIZED.

7.5.1. C-Bindings

TOC

```
OM_uint32 gss_delete_name_attribute(
  OM_uint32                     *minor_status,
  gss_name_t                    name,
  gss_buffer_t                  attr
);
```

TOC

7.6. GSS_Export_name_composite()

Inputs:

- *name NAME

Outputs:

- *major_status INTEGER,
- *minor_status INTEGER,
- *exp_composite_name OCTET STRING

Return major_status codes:

- *GSS_S_COMPLETE indicates no error.
- *GSS_S_FAILURE indicates a general error.

This function outputs a token which can be imported with GSS_Import_name(), using GSS_C_NT_COMPOSITE_EXPORT as the name type and which preserves any name attribute information associated with the input name (which GSS_Export_name() may well not). The token format is no specified here as this facility is intended for inter-process communication only; however, all such tokens MUST start with a two-octet token ID, hex 04 02, in network byte order.
The OID for GSS_C_NT_COMPOSITE_EXPORT is <TBD>.

7.6.1. C-Bindings

TOC

```
OM_uint32 gss_export_name_composite(
  OM_uint32                      *minor_status,
  gss_name_t                     name,
  gss_buffer_t                   exp_composite_name
);
```

TOC

7.7. GSS_Map_name_to_any()

Inputs:

- *name NAME,
- *authenticated BOOLEAN, -- if TRUE only authenticated attributes will be included
- *type_id STRING

Outputs:

- *major_status INTEGER,
- *minor_status INTEGER,
- *output ANY DEFINED BY type_id

Return major_status codes:

- *GSS_S_COMPLETE indicates no error.
- *GSS_S_UNAVAILABLE indicates that the mapping or conversion could not be done. The minor status code may provide additional information.
- *GSS_S_FAILURE indicates a general error. The minor status code may provide additional information.

Whereas name attribute's values are encoded in some network representation applications often require native, language- and/or platform-specific data types. This function provides access to such types.

TOC

7.7.1. C-Bindings

```
typedef struct gss_any *gss_any_t;
OM_uint32 gss_map_name_to_any(
  OM_uint32                      *minor_status,
  gss_name_t                     name,
  int                            authenticated,
  gss_buffer_t                   type_id, // why isn't this 'name'?
  gss_any_t                      output
);
```

Note the new C bindings type, gss_any_t. We define it as a pointer to an incompletely declared struct.

7.8. GSS_Release_any_name_mapping()

Inputs:

- *name NAME,

- *type_id STRING,

- *input ANY DEFINED BY type_id

Outputs:

- *major_status INTEGER,

- *minor_status INTEGER,

Return major_status codes:

- *GSS_S_COMPLETE indicates no error.

- *GSS_S_UNAVAILABLE indicates that the mapping or conversion could not be done. The minor status code may provide additional information.

- *GSS_S_FAILURE indicates a general error. The minor status code may provide additional information.

This function releases, if possible, the objects of language- and/or platform-specific types output by GSS_Map_name_to_any(). If such types have native release functions applications MAY use either those or this function to release the given object.

7.8.1. C-Bindings

TOC

```
typedef struct gss_any *gss_any_t;
OM_uint32 gss_release_any_name_mapping(
  OM_uint32                     *minor_status,
  gss_name_t                    name,
  gss_buffer_t                  type_id,
  gss_any_t                     *input
);
```

8. IANA Considerations

TOC

This document creates a namespace of GSS-API name attributes. Attributes are named by URIs, so no single authority is technically needed for allocation. However future deployment experience may indicate the need for an IANA registry for URIs used to reference names specified by IETF standards. It is expected that this will be a registry of URNs but this document provides no further guidance on this registry.

9. Security Considerations

TOC

This document extends the GSS-API naming model to include support for name attributes. The intention is that name attributes are to be used as a basis for (among other things) authorization decisions or application personalization for applications relying on GSS-API security contexts.
The security of the application may be critically dependent on the security of the attributes. This document classifies attributes as asserted or authenticated. Only authenticated attributes MUST be used if the attribute has security implications for the application (eg authorization decisions) since asserted attributes may easily be controlled by the peer directly.
It is important to understand the meaning of 'authenticated' in this setting. It does not mean that any semantic of the attribute is claimed to be true. The only implication is that a trusted third party has asserted the attribute as opposed to the attribute being asserte by the peer itself. Any additional semantics is always the result of applying

policy. For instance in a given deployment the mail attribute of the subject may be authenticated and sourced from an email system where 'correct' values are kept. In another setting users may be allowed to modify their mail addresses freely. In both cases the 'mail' attribute may be authenticated by virtue of being included in signed SAML attribute assertions lor by other means authenticated by the underlying mechanism.
When the underlying security mechanism does not provide a permanent unique identity (eg anonymous kerberos) the GSS-API naming extensions may be used to provide a replacement permanent unique identity attribute which in this case may be unique for each relying party. This is analogous to the Liberty Alliance targetedID attribute and has similar security implications.

10. Normative References

TOC

[I-D.GSS-NAMING]	Hartman, S., “Desired Enhancements to GSSAPI Naming,” draft-ietf-kitten-gss-naming-01.txt (work in progress), February 2005 (TXT).
[RFC2025]	Adams, C., “The Simple Public-Key GSS-API Mechanism (SPKM),” RFC 2025, October 1996 (TXT).
[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC2743]	Linn, J., “Generic Security Service Application Program Interface Version 2, Update 1,” RFC 2743, January 2000 (TXT).
[RFC2744]	Wray, J., “Generic Security Service API Version 2 : C-bindings,” RFC 2744, January 2000 (TXT).
[RFC3001]	Mealling, M., “A URN Namespace of Object Identifiers,” RFC 3001, November 2000 (TXT).
[RFC4120]	Neuman, C., Yu, T., Hartman, S., and K. Raeburn, “The Kerberos Network Authentication Service (V5),” RFC 4120, July 2005 (TXT).
[RFC5280]	Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” RFC 5280, May 2008 (TXT).

Authors' Addresses

TOC

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

	US
Email:	Nicolas.Williams@sun.com
	Leif Johansson
	Swedish University Network
	Thulegatan 11
	Stockholm
	Sweden
Email:	leifj@sunet.se
URI:	http://www.sunet.se