NETWORK WORKING GROUP
Internet-Draft
Expires: April 19, 2006

N. Williams
Sun

October 16, 2005

Guide to the GSS-APIv3
draft-ietf-kitten-gssapi-v3-guide-to-01.txt

Status of this Memo

Copyright Notice

Abstract

Extensions to the GSS-APIv2 are needed for a number of reasons. This documents describes the extensions being proposed, the resons, possible future directions, and portability, IANA and security considerations. This document does not define any protocol or interface and is purely informational.

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].

2. Introduction

[NOTE: the references section is current fairly empty; the various KITTEN WG work items will be added to this I-D in a subsequent revision.]

Since the advent of the GSS-APIv2 it has come to be used in a number of Internet (and other) protocols and a number of implementations exist. In that time implementors and protocol designers have come to understand both, the GSS-API's strengths, and its shortcommings; we believe now that a number of extensions to the GSS-API are needed. Here these proposed extensions, forming what we may call the GSS-API version 3, are described at a high-level.;

Some of these extensions are intended to facilitate further extensions, so that further major revisions to the GSS-API may not be necessary. Others are intended to fill voids in the the GSS-APIv2.

The extensions being proposed are:

- A pseudo-mechanism OID for the GSS-API itself
- Mechanism attribute inquiry facilities
- Security context extensibility extensions
- Credential extensibility extensions
- Credential export/import

GSS_Store_cred(), for making delegated credentials available for acquisition

Pseudo-mechanism stacking

Naming extensions, to facilitate authorization by identifiers other than names

Additional name types, specifically domain-based naming

A pseudo-random function interface

Channel bindings specifications

Semantic extensions relating to thread- and/or fork-safety

[Have I missed anything? I have a feeling I have. Re-keying?]

...

Additionally, because we foresee future minor extensions, including, specifically, extensions which may impact the various namespaces associated with APIs (symbol names, constant values, class names, etc...) we also propose the establishment of IANA registries for these namespaces.

3. A Pseudo-Mechanism OID for the GSS-API Itself

A mechanism OID is assigned to identify and refer to the GSS-API iself. This is necessary to enable the use of extended inquiry interfaces to inquire about features of a GSS-API implementation specifically, apart from actual mechanisms.

But also, this OID is needed for better error handling, so that minor status codes produced in generic contexts that lack a mechanism OID can be distinguished from minor status codes for a "default" mechanism and properly displayed.

4. Mechanism Attribute Inquiry Facilities

In the course of designing a pseudo-mechanism stacking facility, as well as while considering the impact of all of these extensions on portability, a need for interfaces through which to discover or inquire by features provided by GSS-API mechanisms was discovered.

The proposed mechanism attribute inquiry interfaces consist of:

GSS_Inquire_mech_attrs_for_mech()

GSS_Indicate_mechs_by_mech_attrs()

GSS_Display_mech_attr()

These extensions facilitate portability by allowing GSS-APIv3 applications to discover the features provided by a given implementation of the GSS-API or any mechanisms. These extensions are also useful in facilitating stackable pseudo-mechanisms.

5. Security Context Extensibility Extensions

In order to facilitate future security context options we introduce a GSS_Create_sec_context() interface that creates a security context object, for use with extensions and with GSS_Init_sec_context(), GSS_Accept_sec_context(), and GSS_Inquire_sec_context(). Such security contexts are in a non-established state until they are established through the use of GSS_Init_sec_context() or GSS_Accept_sec_context().

6. Credential Extensibility Extensions

In order to facilitate future extensions to GSS credentials we introduce a GSS_Create_credential(), similar to GSS_Create_sec_context(), interface that creates an "empty" credential.

7. Credential Export/Import

To allow for passing of credentials between different "session contexts," between different hosts, or for storage of post-dated credentials, we introduce a credential export/import facility, much like the security context export/import facility of the GSS-APIv2.

Together with credential extensibility and other extensions this facility may allow for:

Credential delegation at any time

Post-dated credentials, and storage of the such for subsequent use.

...?

8. GSS_Store_cred()

This extension fills a void in the GSS-APIv2 where delegated credentials could not be used except in the context of the same process that received them. With this extension acceptor applications can now make delegated credentials available for use, with GSS_Acquire_cred() et. al., in other process contexts.

[Manipulation of "credential stores" is (may be?) out of scope for this document.]

9. Pseudo-Mechanism Stacking

A number of pseudo-mechanisms are being proposed which are designed to "stack" atop other mechanisms. The possiblities are many, including: a compression mechanism, a perfect forward security mechanism, an many others.

The GSS-APIv2 only had concrete mechanisms and one pseudo-mechanism (SPNEGO) available. With this proposal the mechanism taxonomy is quite expanded:

Concrete mechanisms (e.g., the Kerberos V mechanism)

Composite mechanisms (a concrete mechanism composed with one or more stackable pseudo-mechanisms)

Stackable pseudo-mechanisms

Other pseudo-mechanisms (e.g., SPNEGO, the GSS-API itself)

Although composed mechanisms may be made available for use by GSS-APIv2 applications without any further extensions, use of stackable pseudo-mechanisms can complicate mechanism negotiation; additionally, discovery of mechanisms appropriate for use in one or another context would require hard-coding information about them in GSS-APIv2

applications. Extensions to the GSS-APIv2 could facilitate use of composite.

The mechanism attribute inquiry facilities, together with the forllowing additional interfaces, provide for a complete interface to mechanism composition and for managing the complexity of mechanism negotiation:

GSS_Compose_oid()

GSS_Decompose_oid()

GSS_Release_oid()

GSS_Indicate_negotiable_mechs()

GSS_Negotiate_mechs()

10. Naming Extensions

Some applications make use of exported names, as produced by GSS_Export_name(), to create/manage/evaluate access control lists; we call this name-based authorization.

Exported names typically encode names that are meant for display to humans, not internal identifiers.

In practice this creates a number of problems. E.g., the referential integrity of such access control lists is hard to maintain as principals are added, removed, renamed or old principal names reused.

Additionally, some mechanisms may lack a notion of a "canonical" name for some or all of their principals. Such mechanisms cannot be used by applications that rely on name-based authorization.

<Describe the proposed extensions in this area.>

11. Additional Name Types

<Decribe domain-based names and the need for them.>

12. GSS_Pseudo_random()

<Decribe GSS_Pseudo_random() and the need for it.>

13. Channel Bindings Specifications

14. Semantic and Miscallaneous Extensions

The GSS-APIv2 specifications say nothing about the thread-safety, much less the fork-safety, of the GSS-API. Thread-safety and fork-safety are, after all, platform- and/or language-specific matters. But as support for multi-threading spreads the matter of thread-safety cannot be avoided. The matter of fork-safety is specific to platforms that provide a "fork()" function, or similar.

<describe the GSS-APIv3's thread-safety requirements>

<reference the portability considerations section>

15. Portability Considerations

The potential for additional generic, mechanism-specific, language binding-specific and, most importantly, semantic extensions to the GSS-APIv3 may create application portability problems. The mechanism attribute inquiry facilities of the GSS-APIv3 and the pseudo-mechanism OID for the GSS-API itself double as a run-time facility for discovery of feature availability. Run-time feature discovery facilities, in turn, can be used at application build-time as well by building small applications to display the available features.

<...>

16. IANA Considerations

<Describe the namespace issues associated with future minor extensions to the GSS-APIv3 and the IANA registries to be created to cope with them.>

17. Security Considerations

<...>

18. Normative

[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

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in

this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.