Kitten Working Group
Internet-Draft
Obsoletes: RFC 2831 (if approved)
(if approved)
Intended status: Informational
Expires: October 24, 2011

A. Melnikov
Isode Limited
April 22, 2011

Moving DIGEST-MD5 to Historic
draft-ietf-kitten-digest-to-historic-04

Abstract

This memo describes problems with the DIGEST-MD5 Simple Authentication and Security Layer (SASL) mechanism as specified in RFC 2831. It marks DIGEST-MD5 as OBSOLETE in the IANA Registry of SASL mechanisms, and moves RFC 2831 to Historic. status.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on October 24, 2011.

Copyright Notice

Table of Contents

1. Overview

[RFC2831] defined how HTTP Digest Authentication [RFC2617] can be used as a Simple Authentication and Security Layer (SASL) [RFC4422] mechanism for any protocol that has a SASL profile. It was intended both as an improvement over CRAM-MD5 [RFC2195] and as a convenient way to support a single authentication mechanism for web, email, LDAP, and other protocols. While it can be argued that it was an improvement over CRAM-MD5, many implementors commented that the additional complexity of DIGEST-MD5 made it difficult to implement fully and securely.

Below is an incomplete list of problems with DIGEST-MD5 mechanism as specified in RFC 2831:

1. The mechanism had too many options and modes. Some of them were not well described and were not widely implemented. For example, DIGEST-MD5 allowed the "qop" directive to contain multiple values, but it also allowed for multiple qop directives to be specified. The handling of multiple options was not specified, which resulted in minor interoperability problems. Some implementations amalgamated multiple qop values into one, while others treated multiple qops as an error. Another example is the use of an empty authorization identity. In SASL an empty authorization identity means that the client is willing to authorize as the authentication identity. The document was not clear on whether the authzid must be omitted or can be specified with the empty value to convey this. The requirement for backward compatibility with HTTP Digest meant that the situation was even worse. For example DIGEST-MD5 required all usernames/ passwords which can be entirely represented in ISO-8859-1 charset to be down converted from UTF-8 to ISO-8859-1. Another example is use of quoted strings. Handling of characters that needed escaping was not properly described and the DIGEST-MD5 document had no examples to demonstrate correct behavior.

2. The document used ABNF from RFC 822 [RFC0822], which allows an extra construct and allows for "implied folding whitespace" to be inserted in many places. The difference from ABNF [RFC5234] was confusing for some implementors. As a result, many implementations didn’t accept folding whitespace in many places where it was allowed.

3. The DIGEST-MD5 document uses the concept of a "realm" to define a collection of accounts. A DIGEST-MD5 server can support one or more realms. The DIGEST-MD5 document didn’t provide any guidance on how realms should be named, and, more importantly, how they can be entered in User Interfaces (UIs). As the result many

DIGEST-MD5 clients had confusing UIs, didn't allow users to enter a realm and/or didn't allow users to pick one of the server supported realms.

4. Use of username in the inner hash. The inner hash of DIGEST-MD5 is an MD5 hash of colon separated username, realm and password. Implementations may choose to store inner hashes instead of clear text passwords. While this has some useful properties, such as protection from compromise of authentication databases containing the same username and password on other servers, if a server with the username and password is compromised, however this was rarely done in practice. Firstly, the inner hash is not compatible with widely deployed Unix password databases, and second, changing the username would invalidate the inner hash.

5. Description of DES/3DES [DES] and RC4 security layers are inadequate to produce independently-developed interoperable implementations. In the DES/3DES case this was partly a problem with existing DES APIs.

6. DIGEST-MD5 outer hash (the value of the "response" directive) didn't protect the whole authentication exchange, which made the mechanism vulnerable to "man in the middle" (MITM) attacks, such as modification of the list of supported qops or ciphers.

7. The following features are missing from DIGEST-MD5, which make it insecure or unsuitable for use in protocols:

   A. Lack of channel bindings [RFC5056].

   B. Lack of hash agility (i.e. no easy way to replace the MD5 hash function with another one).

   C. Lack of support for SASLPrep [RFC4013] or any other type of Unicode character normalization of usernames and passwords. The original DIGEST-MD5 document predates SASLPrep and doesn't recommend any Unicode character normalization.

8. The cryptographic primitives in DIGEST-MD5 are not up to today's standards, in particular:

   A. The MD5 hash is sufficiently weak to make a brute force attack on DIGEST-MD5 easy with common hardware [RFC6151].

   B. Using the RC4 algorithm for the security layer without discarding the initial key stream output is prone to attack [RC4].

C. The DES cipher for the security layer is considered insecure due to its small key space [RFC3766].

Note that most of the problems listed above are already present in the HTTP Digest authentication mechanism.

Because DIGEST-MD5 was defined as an extensible mechanism, it would be possible to fix most of the problems listed above. However this would increase implementation complexity of an already complex mechanism even further, so the effort would not be worth the cost. In addition, an implementation of a "fixed" DIGEST-MD5 specification would likely either not interoperate with any existing implementation of RFC 2831, or would be vulnerable to various downgrade attacks.

Note that despite DIGEST-MD5 seeing some deployment on the Internet, this specification recommends obsoleting DIGEST-MD5 because DIGEST-MD5, as implemented, is not a reasonable candidate for further standardization and should be deprecated in favor of one or more new password-based mechanisms currently being designed.

The SCRAM family of SASL mechanisms [RFC5802] has been developed to provide similar features as DIGEST-MD5 but with a better design.

2. Security Considerations

Security issues are discussed through out this document.

3. IANA Considerations

IANA is requested to change the "Intended usage" of the DIGEST-MD5 mechanism registration in the SASL mechanism registry to OBSOLETE. The SASL mechanism registry is specified in [RFC4422] and is currently available at:

http://www.iana.org/assignments/sasl-mechanisms

4. Acknowledgements

The author gratefully acknowledges the feedback provided by Chris Newman, Simon Josefsson, Kurt Zeilenga, Sean Turner and Abhijit Menon-Sen. Various text was copied from other RFCs, in particular from RFC 2831.

5. References

5.1. Normative References

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[RFC2831] Leach, P. and C. Newman, "Using Digest Authentication as a SASL Mechanism", RFC 2831, May 2000.

5.2. Informative References

[DES] National Institute of Standards and Technology, "Data Encryption Standard (DES)", FIPS PUB 46-3, October 1999.

[RC4] Strombergson, J. and S. Josefsson, "Test vectors for the stream cipher RC4", draft-josefsson-rc4-test-vectors-02.txt (work in progress), June 2010.

[RFC0822] Crocker, D., "Standard for the format of ARPA Internet text messages", STD 11, RFC 822, August 1982.

[RFC2195] Klensin, J., Catoe, R., and P. Krumviede, "IMAP/POP AUTHorize Extension for Simple Challenge/Response", RFC 2195, September 1997.

[RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, April 2004.

[RFC4013] Zeilenga, K., "SASLprep: Stringprep Profile for User Names and Passwords", RFC 4013, February 2005.

[RFC4422] Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, November 2007.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5802] Newman, C., Menon-Sen, A., Melnikov, A., and N. Williams, "Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms", RFC 5802, July 2010.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, March 2011.

Author's Address

Alexey Melnikov
Isode Limited
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

Email: Alexey.Melnikov@isode.com
URI: http://www.melnikov.ca/

NETWORK WORKING GROUP
Internet-Draft
Intended status: Standards Track
Expires: October 1, 2017

N. Williams
Cryptonector LLC
A. Melnikov
Isode Ltd

March 30, 2017

Namespace Considerations and Registries for GSS-API Extensions
draft-ietf-kitten-gssapi-extensions-iana-11.txt

Abstract

This document describes the ways in which the GSS-API may be extended and directs the creation of an IANA registry for various GSS-API namespaces.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on October 1, 2017.

Copyright Notice

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

There is a need for private-use and mechanism-specific extensions to the Generic Security Services Application Programming Interface (GSS-API). As such extensions are designed and standardized (or not), both at the IETF and elsewhere, there is a non-trivial risk of namespace pollution and conflicts. To avoid this we set out guidelines for extending the GSS-API and direct the creation of an IANA registry for GSS-API namespaces.

Registrations of individual items and sub-namespaces are allowed. Each sub-namespace may provide different rules for registration, e.g., for mechanism-specific and private-use extensions.

3. Extensions to the GSS-API

Extensions to the GSS-API can be categorized as follows:

- Abstract API extensions

- Implementation-specific
- Mechanism-specific
- Language binding-specific

Extensions to the GSS-API may be purely semantic, without effect on the GSS-API's namespaces. Or they may introduce new functions, constants, types, etc...; these clearly affect the GSS-API namespaces.

Extensions that affect the GSS-API namespaces should be registered with the IANA as described herein.

4. Generic GSS-API Namespaces

The abstract API namespaces for the GSS-API are:

- Type names
- Function names
- Constant names for various types
- Constant values for various types
- Name types (OID, type name and syntaxes)

Additionally we have namespaces associates with the OBJECT IDENTIFIER (OID) type. The IANA already maintains a registry of such OIDs:

- Mechanism OIDs
- Name Type OIDs

5. Language Binding-Specific GSS-API Namespaces

Language binding specific namespaces include, among others:

- Header/interface module names
- Object classes and/or types
- Methods and/or functions
- Constant names
- Constant values

6. Extension-Specific GSS-API Namespaces

Extensions to the GSS-API may create additional namespaces. See Section 8.2.

7. Registration Form

Registrations for GSS-API namespaces SHALL take the following form:

Registration Field	Possible Values	Description
Bindings	'Generic', 'C-bindings', 'Java', 'C#', <programming language name>	Indicates the name of the programming language that this registration involves, or, if 'Generic', that this is an entry for the generic abstract GSS-API (i.e., not specific to any programming language).
Registration type	'Instance', 'Sub-Namespace'	Indicates whether this entry reserves a given symbol name (and possibly, constant value), or whether it reserves an entire sub-namespace (the name is a pattern) or constant value range.
Object Type	<Symbol> defined by the binding language (for example 'Data-Type', 'Function', 'Method', 'Integer', 'String', 'OID', 'Context-Flag', 'Name-Type', 'Macro', 'Header-File-Name', 'Module-Name', 'Class')	Indicates the type of the object whose symbolic name or constant value this entry registers. The possible values of this field depend on the programming language in question, therefore they are not all specified here.
Symbol Name/Prefix	<Symbol name or name pattern>	The name of a symbol or symbol sub-namespace being

		registered. See Section 8.2.1
Binding of	<Name of abstract API element of which this object is a binding>	If the registration is for a specific language binding of the GSS-API, then this names the abstract API element of which it is a binding (OPTIONAL).
Constant Value/Range	<Constant value> or <constant value range>	The value of the constant named by the <Symbol Name/Prefix>. This field is present only for Instance and Sub-namespace registrations of Constant object types.
Description	<Text>	Description of the registration. Multiple instances of this field may result (see Section 8.2.3).
Registration Rules	<Reference> to an IANA registration Policy defined in [RFC5226] (or an RFC that updates it), for instance 'IESG Approval', 'Expert Review', 'First Come First Served', 'Private Use'.	Describes the rules for allocation of items that fall in this sub-namespace, for entries with Registration Type of Sub-namespace (OPTIONAL). For private use sub-namespaces the submitter MUST provide the e-mail address of a responsible contact. If this field is not specified for a sub-namespace, the default registration rules specified in Section 8.2 apply.
Reference	<Reference>	Reference to a document that describes the registration, if any (OPTIONAL). Multiple instances of this field are allowed, with one reference each.
Expert Reviewer	<Name of expert reviewers, possibly	OPTIONAL, see Section 8.2.2. Multiple instances of this

	WG names>	field are allowed, with one expert reviewer per-instance. Leave this field blank when requesting a registration. It will be filled in by the Expert who reviews the registration.
Expert Review Notes	<Notes from the expert review>	Expert reviewers may request that some comments be included with the registration, e.g., regarding security considerations of the registered extension.
Status	'Registered' or 'Obsoleted'	Status of the registration.
Obsoleting Reference	<Reference>	Reference to a document, if any, that obsoletes this registration. Multiple instances of this field are allowed, with one reference each. (OPTIONAL)

The IANA should create a single GSS-API namespace registry, or multiple registries, one for symbolic names and one for constant values, and/or it may create a registry per-programming language, at its convenience.

Entries in these registries should consist of all the fields from their corresponding registration entries.

Entries should be sorted by: programming language, registration type, object type, and symbol name/pattern.

8. IANA Considerations

This document deals with IANA considerations throughout. Specifically it creates a single registry of various kinds of things, though the IANA may instead create multiple registries, each for one of those kinds of things. Of particular interest may be that IANA will now be the registration authority for the GSS-API name type OID space.

8.1. Initial Namespace Registrations

Initial registry content corresponding to the items defined in [RFC2743], [RFC2744], [RFC2853], [RFC1964] and [RFC4121] and others will be supplied during the IANA review portion of the RFC publishing process. [[Note to RFC Editor: Delete the following sentence before publication:]] The KITTEN WG chairs MUST indicate that such content has been reviewed by the WG and that there is WG consensus that the entries are in agreement with those RFCs.

8.1.1. Example registrations

In order to sanity check recommended IANA registration templates, this section registers several entries.

Registration Field	Possible Values
Bindings	C-bindings
Registration type	Instance
Object Type	Function
Symbol Name	gss_init_sec_context
Binding of	GSS_Init_sec_context
Constant Value/Range	N/A
Description	Create a security context by initiator
Registration Rules	N/A
Reference	RFC 2744
Expert Reviewer	Kitten WG
Expert Review Notes	
Status	Registered
Obsoleting Reference	N/A

Registration Field	Possible Values
Bindings	C-bindings
Registration type	Instance
Object Type	Function
Symbol Name	gss_accept_sec_context
Binding of	GSS_Accept_sec_context
Constant Value/Range	N/A
Description	Accept a security context from initiator
Registration Rules	N/A
Reference	RFC 2744
Expert Reviewer	Kitten WG
Expert Review Notes	
Status	Registered
Obsoleting Reference	N/A

Registration Field	Possible Values
Bindings	C-bindings
Registration type	Instance
Object Type	Context-Flag
Symbol Name	GSS_C_DELEG_FLAG
Binding of	deleg_state or deleg_req_flag
Constant Value/Range	1
Description	On output (if set): Delegated credentials are available via the delegated_cred_handle parameter of GSS_Accept_sec_context. On input (if set): With the call to GSS_Init_sec_context, delegate credentials to the acceptor.
Registration Rules	N/A
Reference	RFC 2744
Expert Reviewer	Kitten WG
Expert Review Notes	
Status	Registered
Obsoleting Reference	N/A

8.2. Registration Maintenance Guidelines

Standards-Track RFCs can create new items with any non-conflicting Symbol Name/Prefix value for this registry by virtue of IESG approval to publish as a Standards-Track RFC -- that is, without additional expert review.

Standards-Track RFCs can mark existing entries as obsolete, and can even create conflicting entries if explicitly stated (the IESG, of course, should review conflicts carefully, and may reject them).

IANA shall also consider submissions from individuals, and via Informational and Experimental RFCs, subject to Expert Review. IANA SHALL allow such registrations if a) they are not conflicting, b) provided that the registration is for object types other than Context-Flags, and c) subject to expert review. Guidelines for expert reviews are given below.

8.2.1. Sub-Namespace Symbol Pattern Matching

Sub-namespace registrations must provide a pattern for matching symbols for which the sub-namespace's registration rules apply. The pattern consists of a string with the following special tokens:

- '*', meaning "match any string."
- "%m", meaning "match any mechanism family short-hand name."
- "%i", meaning "match any implementor vanity short-hand name."

For example, "GSS_%m_*" matches "GSS_krb5_foo" since "krb5" is a common short-hand for the Kerberos V GSS-API mechanism [RFC1964]. But "GSS_%m_*" does not match "GSS_foo_bar" unless "foo" is asserted to be a short-hand for some mechanism.

8.2.2. Expert Reviews of Individual Submissions

[[The following paragraph should be deleted from the document before publication, as it will not age well. It should be moved to the shepherding write-up.]]

Expert review selection SHALL be done as follows. If, at the time that the IANA receives an individual submission for registration in this registry, there are any IETF Working Groups chartered to produce GSS-API-related documents, then the IANA SHALL ask the chairs of such WGs to be expert reviewers or to name one. If there are no such WGs at that time, then the IANA SHALL ask past chairs of the KITTEN WG and the author/editor of this RFC to act as expert reviewers or name an alternate.

Expert reviewers of individual registration submissions with Registration Type == Sub-namespace should check that the registration request has a suitable description (which doesn't need to be sufficiently detailed for others to implement) and that the Symbol Name/Prefix is sufficiently descriptive of the purpose of the sub-namespace or reflective of the name of the submitter or associated company.

Expert reviewers of individual registration submissions with

Registration Type == Instance should check that the Symbol Name falls under a sub-namespace controlled by the submitter. Registration of such entries which do not fall under such a sub-namespace may be allowed provided that they correspond to long existing non-standard extensions to the GSS-API and this can be easily checked or demonstrated, otherwise IESG Protocol Action is REQUIRED (see previous section). Also, reviewers should check that any registration of constant values have a detailed description that is suitable for other implementors to reproduce, and that they don't conflict with other usages or are otherwise dangerous in the reviewers estimation.

Expert reviewers should review impact on mechanisms, security and interoperability, and may reject or annotate registrations which can have mechanism impact that requires IESG protocol action. Consider, for example, new versions of GSS_Init_sec_context() and/or GSS_Accept_sec_context which have new input and/or output parameters which imply changes on the wire or in behaviour that may result in interoperability issues. A reviewer could choose to add notes to the registration describing such issues, or the reviewer might conclude that the danger to Internet interoperability is sufficient to warrant rejecting the registration.

8.2.3. Change Control

Registered entries may be marked obsoleted using the same expert review process as for registering new entries. Obsoleted entries are not, however, to be deleted, but merely marked having Obsoleted Status. Note that entries may be created as obsoleted to record the fact that the given symbol(s) have been used before, even though continued use of them is discouraged.

Registered entries may also be updated in two other ways: additional references, obsoleting references, and descriptions may be added.

All changes are subject to expert review, except for changes to registrations in a sub-namespace which are subject to the rules of the relevant sub-namespace. The submitter of a change request need not be the same as the original submitter.

Registrations may be modified by addition, but under no circumstance may any fields be modified except for the Status field or Contact Address, or to correct for transcription errors in filing or processing registration requests.

The IANA SHALL add a field describing the date that a an addition or modification was made, and a description of the change.

9. Security Considerations

General security considerations relating to IANA registration services apply; see [RFC5226].

Also, expert reviewers should look for and may document security related issues with submitters' GSS-API extensions, to the best of the reviewers' ability given the information furnished by the submitter. Reviewers may add comments regarding their limited ability to review a submission for security problems if the submitter is unwilling to provide sufficient documentation.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, DOI 10.17487/RFC2743, January 2000, <http://www.rfc-editor.org/info/rfc2743>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

10.2. Informative References

[RFC1964] Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, DOI 10.17487/RFC1964, June 1996, <http://www.rfc-editor.org/info/rfc1964>.

[RFC2744] Wray, J., "Generic Security Service API Version 2 : C-bindings", RFC 2744, DOI 10.17487/RFC2744, January 2000, <http://www.rfc-editor.org/info/rfc2744>.

[RFC2853] Kabat, J. and M. Upadhyay, "Generic Security Service API Version 2 : Java Bindings", RFC 2853, DOI 10.17487/RFC2853, June 2000, <http://www.rfc-editor.org/info/rfc2853>.

[RFC4121] Zhu, L., Jaganathan, K., and S. Hartman, "The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2", RFC 4121, DOI 10.17487/RFC4121, July 2005, <http://www.rfc-editor.org/info/rfc4121>.

Authors' Addresses

Nicolas Williams
Cryptonector LLC

Email: nico@cryptonector.com

Alexey Melnikov
Isode Ltd
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

Email: Alexey.Melnikov@isode.com

KITTEN WORKING GROUP
Internet-Draft
Intended status: Standards Track
Expires: December 2, 2012

N. Williams
Cryptonector, LLC
L. Johansson
SUNET
S. Hartman
Painless Security
S. Josefsson
SJD AB

May 31, 2012

GSS-API Naming Extensions
draft-ietf-kitten-gssapi-naming-exts-15

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.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on December 2, 2012.

Copyright Notice

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

As described in [RFC4768] the GSS-API's naming architecture suffers from certain limitations. This document defines concrete GSS-API extensions.

A number of extensions to the GSS-API [RFC2743] and its C Bindings [RFC2744] 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 (e.g., 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.

Future updates of this specification may involve adding an attribute namespace for attributes that only have application-specific semantics. Note that mechanisms will still need to know how to transport such attributes. The IETF may also wish to add functions by which to inquire whether a mechanism(s) understands a given attribute name or namespace, and to list which attributes or attribute namespaces a mechanism understands. Finally, the IETF may want to consider adding a function by which to determine the name of the issuer of a name attribute.

3. Name Attribute Authenticity

An attribute is 'authenticated' if and only if 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 (e.g., 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 was authorized to assert the attribute. Such interpretations SHOULD be the result of applying local policy to the attribute.

An un-authenticated attribute is called _asserted_ in what follows. This is not to be confused with other uses of the word asserted or assertion such as "SAML attribute assertion", the attributes of which may be authenticated in the sense of this document for instance if the SAML attribute assertion was signed by a key trusted by the peer.

4. Name Attributes/Values as ACL Subjects

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.

5. Naming Contexts

Several factors influence the context in which a name attribute is interpreted. One is the trust context.

As discussed previously, applications apply local policy to determine whether a particular peer credential issuer is trusted to make a given statement. Different GSS-API mechanisms and deployments have different trust models surrounding attributes they provide about a name.

For example, Kerberos deployments in the enterprise typically trust a KDC to make any statement about principals in a realm. This includes attributes such as group membership.

In contrast, in a federated SAML environment, the identity provider typically exists in a different organization than the acceptor. In this case, the set of group memberships or entitlements that the IDP is permitted to make needs to be filtered by the policy of the

acceptor and federation.

So even an attribute containing the same information such as e-mail address would need to be treated differently by the application in the context of an enterprise deployment from the context of a federation.

Another aspect related to trust is the role of the credential issuer in providing the attribute. Consider Kerberos PKINIT [RFC4556]. In this protocol, a public key and associated certificate are used to authenticate to a Kerberos KDC. Consider how attributes related to a pkinit certificate should be made available in GSS-API authentications based on the Kerberos ticket. In some deployments the certificate may be fully trusted; in including the certificate information in the ticket, the KDC permits the acceptor to trust the information in the certificate just as if the KDC itself had made these statements. In other deployments, the KDC may have authorized a hash of the certificate without evaluating the content of the certificate or generally trusting the issuing certification authority. In this case, if the certificate were included in the issued ticket, the KDC would only be making the statement that the certificate was used in the authentication. This statement would be authenticated, but would not imply that the KDC stated particular attributes of the certificate described the initiator.

Another aspect of context is encoding of the attribute information. An attribute containing an ASCII [ANSI.X3-4.1986] or UTF-8 [RFC3629] version of an e-mail address could not be interpreted the same as a ASN.1 Distinguished Encoding Rules e-mail address in a certificate.

All of these contextual aspects of a name attribute affect whether two attributes can be treated the same by an application and thus whether they should be considered the same name attribute. In the GSS-API naming extensions, attributes that have different contexts MUST have different names so they can be distinguished by applications. As an unfortunate consequence of this requirement, multiple attribute names will exist for the same basic information. That is, there is no single attribute name for the e-mail address of an initiator. Other aspects of how mechanisms describe information about subjects would already make this true. For example, some mechanisms use OIDs to name attributes; others use URIs.

Local implementations or platforms are likely to have sufficient policy and information to know when contexts can be treated as the same. For example the GSS-API implementation may know that a particular certification authority can be trusted in the context of a pkinit authentication. The local implementation may have sufficient policy to know that a particular credential issuer is trusted to make

a given statement. In order to take advantage of this local knowledge within the GSS-API implementation, naming extensions support the concept of local attributes in addition to standard attributes. For example, an implementation might provide a local attribute for e-mail address. The implementation would specify the encoding and representation of this attribute; mechanism-specific standards attributes would be re-encoded if necessary to meet this representation. Only e-mail addresses in contexts that meet the requirements of local policy would be mapped into this local attribute.

Such local attributes inherently expose a tradeoff between interoperability and usability. Using a local attribute in an application requires knowledge of the local implementation. However using a standardized attribute in an application requires more knowledge of policy and more validation logic in the application. Sharing this logic in the local platform provides more consistency across applications as well as reducing implementation costs. Both options are needed.

6. Representation of Attribute Names

Different underlying mechanisms (e.g., SAML or X.509 certificates) provide different representations for the names of their attribute. In X.509 certificates, most objects are named by object identifiers (OIDs). The type of object (certificate extension, name constraint, keyPurposeID, etc) along with the OID is sufficient to identify the attribute. By contrast, according to Section 8.2 and 2.7.3.1 of [OASIS.saml-core-2.0-os], the name of an attribute has two parts. The first is a URI describing the format of the name. The second part, whose form depends on the format URI, is the actual name. In other cases an attribute might represent a certificate that plays some particular role in a GSS-API mechanism; such attributes might have a simple mechanism-defined name.

Attribute names MUST support multiple components. If there are more than one component in an attribute name, the more significant components define the semantics of the less significant components.

Attribute names are represented as OCTET 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).

If an attribute name contains a space (ASCII 0x20), the first space separates the most significant or primary component of the name from the remainder. We may refer to the primary component of the

attribute name as the attribute name's "prefix". If there is no space, the primary component is the entire name, otherwise it defines the interpretation of the remainder of the name.s

If the primary component contains an ASCII : (0x3a), then the primary component is a URI. Otherwise, the attribute is a local attribute and the primary component has meaning to the implementation of GSS-API or to the specific configuration of the application. Local attribute names with an at-sign ('@') in them are reserved for future allocation by the IETF.

Since attribute names are split at the first space into prefix and suffix, there is a potential for ambiguity if a mechanism blindly passes through a name attribute whose name it does not understand. In order to prevent such ambiguities the mechanism MUST always prefix raw name attributes with a prefix that reflects the context of the attribute.

Local attribute names under the control of an administrator or a sufficiently trusted part of the platform need not have a prefix to describe context.

7. API

7.1. SET OF OCTET STRING

The construct SET OF OCTET STRING occurs once in RFC 2743 [RFC2743] where it is used to represent a set of status strings in the GSS_Display_status call. The Global Grid Forum has defined SET OF OCTET STRING as a buffer-set type in GFD.024 [GFD.024] which also provides one API for memory management of these structures. The normative reference to GFD.024 [GFD.024] is for the buffer set functions defined in section 2.5 and the associated buffer set C types defined in section 6 (namely gss_buffer_set_desc, gss_buffer_set_t, gss_create_empty_buffer_set, gss_add_buffer_set_member, gss_release_buffer_set). Nothing else from GFD.024 is required to implement this document. In particular, that document specify changes in behaviour existing GSS-API functions in section 3: implementing those changes are not required to implement this document. Any implementation of SET OF OCTET STRING for use by this specification MUST preserve order.

7.2. Const types

The C bindings for the new APIs uses some types from [RFC5587] to avoid issues with the use of "const". The normative reference to [RFC5587] is for the C types specified in Figure 1 of 3.4.6, nothing

else from that document is required to implement this document.

7.3. GSS_Display_name_ext()

Inputs:

- name INTERNAL NAME,
- display_as_name_type OBJECT IDENTIFIER

Outputs:

- major_status INTEGER,
- minor_status INTEGER,
- display_name OCTET STRING -- caller must release with GSS_Release_buffer()

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 Mechanism Names (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.

7.3.1. C-Bindings

The display_name buffer is de-allocated by the caller with gss_release_buffer.

```
OM_uint32 gss_display_name_ext(
  OM_uint32                    *minor_status,
  gss_const_name_t             name,
  gss_const_OID                display_as_name_type,
  gss_buffer_t                 display_name
);
```

7.4. GSS_Inquire_name()

Inputs:

- name INTERNAL NAME

Outputs:

- major_status INTEGER,
- minor_status INTEGER,
- name_is_MN BOOLEAN,
- mn_mech OBJECT IDENTIFIER,
- attrs SET OF OCTET STRING -- the caller is responsible for de-allocating memory using GSS_Release_buffer_set

Return major_status codes:

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

This function outputs the set of attributes of a name. It also indicates if a given name is an Mechanism Name (MN) or not and, if it is, what mechanism it's an MN of.

7.4.1. C-Bindings

```
OM_uint32 gss_inquire_name(
  OM_uint32                    *minor_status,
  gss_const_name_t             name,
  int                          *name_is_MN,
  gss_OID                      *MN_mech,
  gss_buffer_set_t             *attrs
);
```

The gss_buffer_set_t is used here as the C representation of SET OF OCTET STRING. This type is used to represent a set of attributes and is a NULL-terminated array of gss_buffer_t. The gss_buffer_set_t type and associated API is defined in GFD.024 [GFD.024]. The "attrs" buffer set is de-allocated by the caller using gss_release_buffer_set().

7.5. GSS_Get_name_attribute()

Inputs:

- name INTERNAL NAME,
- attr OCTET STRING

Outputs:

- major_status INTEGER,
- minor_status INTEGER,
- authenticated BOOLEAN, -- TRUE if and only if authenticated by the trusted peer credential source.
- complete BOOLEAN -- TRUE if and only if this represents a complete set of values for the name.
- values SET OF OCTET STRING -- the caller is responsible for de-allocating memory using GSS_Release_buffer_set.
- display_values SET OF OCTET STRING -- the caller is responsible for de-allocating memory using GSS_Release_buffer_set

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.

7.5.1. C-Bindings

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_get_name_attribute() 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. The status of the complete and authenticated flags MUST NOT change between multiple calls to iterate over values for an attribute.

The output buffers "value" and "display_value" are de-allocated by the caller using gss_release_buffer().

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

7.6. GSS_Set_name_attribute()

Inputs:

- name INTERNAL NAME,
- complete BOOLEAN, -- TRUE if and only if this represents a complete set of values for the name.

- attr OCTET 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 NAME is not known or could not be set.
- GSS_S_FAILURE indicates a general error.

When the given NAME object is an MN this function MUST fail (with GSS_S_FAILURE) if the mechanism for which the name is an MN does not recognize the attribute name or the namespace it belongs to. This is because name attributes generally have some semantics that mechanisms must understand.

On the other hand, when the given name is not an MN this function MAY succeed even if none of the available mechanisms understand the given attribute, in which subsequent credential acquisition attempts (via GSS_Acquire_cred() or GSS_Add_cred()) with the resulting name MUST fail for mechanisms that do not understand any one or more name attributes set with this function. Applications may wish to use a non-MN, then acquire a credential with that name as the desired name. The acquired credentials will have elements only for the mechanisms that can carry the name attributes set on the name.

Note that this means that all name attributes are locally critical: the mechanism(s) must understand them. The reason for this is that name attributes must necessarily have some meaning that the mechanism must understand, even in the case of application-specific attributes (in which case the mechanism must know to transport the attribute to any peer). However, there is no provision to ensure that peers understand any given name attribute. Individual name attributes may be critical with respect to peers, and the specification of the attribute will have to indicate which of the mechanism’s protocol or the application is expected to enforce criticality.

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.

7.6.1. 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_const_name_t             name,
  int                          complete,
  gss_const_buffer_t           attr,
  gss_const_buffer_t           value
);
```

7.7. GSS_Delete_name_attribute()

Inputs:

- name INTERNAL NAME,
- attr 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 NAME is not known.
- GSS_S_UNAUTHORIZED indicates that a forbidden delete operation was attempted, such as 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.

7.7.1. C-Bindings

```
OM_uint32 gss_delete_name_attribute(
  OM_uint32                    *minor_status,
  gss_const_name_t             name,
  gss_const_buffer_t           attr
);
```

7.8. GSS_Export_name_composite()

Inputs:

- name INTERNAL NAME

Outputs:

- major_status INTEGER,
- minor_status INTEGER,
- exp_composite_name OCTET STRING -- the caller is responsible for de-allocating memory using GSS_Release_buffer

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 (including the authenticated/complete flags) associated with the input name (which GSS_Export_name() may well not). The token format is not 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>.

7.8.1. C-Bindings

The "exp_composite_name" buffer is de-allocated by the caller with gss_release_buffer.

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

8. IANA Considerations

This specification has no actions for IANA.

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

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 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. Asserted (non-authenticated) attributes MUST NOT be used if the attribute has security implications for the application (e.g., 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. Authenticated does not imply 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 asserted by the peer itself. Any additional semantics are 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 'authoritive' values are kept. In another situation 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 or by other means authenticated by the underlying mechanism.

When the underlying security mechanism does not provide a permanent unique identity (e.g., anonymous kerberos), GSS-API naming extensions may be used to provide a permanent unique identity attribute. This may be a globally unique identifier, a value unique within the namespace of the attribute issuer, or a "directed" identifier that is unique per peer acceptor identity. SAML, to use one example technology, offers a number of built-in constructs for this purpose, such as a <NameID> with a Format of "urn:oasis:names:tc:SAML:2.0:nameid-format:persistent". SAML deployments also typically make use of domain-specific attribute types that can serve as identifiers.

10. References

10.1. Normative References

[GFD.024] Argonne National Laboratory, National Center for Supercomputing Applications, Argonne National Laboratory, and Argonne National Laboratory, "GSS-API Extensions", GFD GFD.024, June 2004.

[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.

[RFC5587] Williams, N., "Extended Generic Security Service Mechanism Inquiry APIs", RFC 5587, July 2009.

10.2. Informative References

[ANSI.X3-4.1986]

American National Standards Institute, "Coded Character Set - 7-bit American Standard Code for Information Interchange", ANSI X3.4, 1986.

[OASIS.saml-bindings-2.0-os]
Cantor, S., Hirsch, F., Kemp, J., Philpott, R., and E. Maler, "Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-bindings-2.0-os, March 2005.

[OASIS.saml-core-2.0-os]
Cantor, S., Kemp, J., Philpott, R., and E. Maler, "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-core-2.0-os, March 2005.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4556] Zhu, L. and B. Tung, "Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)", RFC 4556, June 2006.

[RFC4768] Hartman, S., "Desired Enhancements to Generic Security Services Application Program Interface (GSS-API) Version 3 Naming", RFC 4768, December 2006.

Authors' Addresses

Nicolas Williams
Cryptonector, LLC

Email: nico@cryptonector.com

Leif Johansson
Swedish University Network
Thulegatan 11
Stockholm
Sweden

Email: leifj@sunet.se
URI: http://www.sunet.se

Sam Hartman
Painless Security

Phone:
Fax:
Email: hartmans-ietf@mit.edu
URI:

Simon Josefsson
SJD AB
Hagagatan 24
Stockholm 113 47
SE

Email: simon@josefsson.org
URI: http://josefsson.org/

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 13, 2011

T. Yu
MIT Kerberos Consortium

July 12, 2010

Desired changes to the GSS-API
draft-yu-kitten-api-wishlist-01

Abstract

Feedback from GSS-API implementors and application developers suggests that the API as it currently exists would benefit from improvements. This memo collects some specific suggestions of KITTEN WG participants.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on January 13, 2011.

Copyright Notice

Table of Contents

1. Introduction

Experiences of GSS-API implementors and GSS-API application developers, particularly with the C bindings, suggest that the GSS-API would benefit from certain improvements. Some of these suggestions collected from the KITTEN working group include:

1. initialization/new credentials
2. listing/iterating credentials
3. exporting/importing credentials
4. error message reporting
5. asynchronous calls
6. security strength reporting
7. programmer friendliness

This summary is not complete; it is meant as a starting point.

2. Asynchronous calls

The desire for supporting asynchronous calls is a specific case of a generally accepted goal of increasing concurrency. Proponents of this goal typically note that new computers appear to be gaining more processor cores faster than they are gaining computing speed per core. Asynchronous calls (or event-based solutions) are an alternative to the traditional multi-threading model for increasing concurrency.

The existing C bindings say nothing about thread safety for the GSS-API. Implementors have considered various interpretations of thread safety, including using internal mutex locks within a GSS-API implementation to provide thread safety for callers. While the existing C bindings allow for such an approach, the traditional threaded programming model has its drawbacks.

In addition, some GSS-API mechanisms are nearly impossible to implement in a way that prevents gss_init_sec_context and such from blocking on I/O operations, particularly network I/O. An application that attempts to achieve high concurrency must dedicate a thread for each context establishment operation, for example. This can be problematic on platforms where threads are expensive.

Forcing callers to call into an event loop provided by GSS-API is not desirable.

3. Error message reporting

Existing GSS-API facilities for obtaining error information are limited to 32-bit major and minor status codes. This prevents callers from obtaining detailed (perhaps textual) information that may assist in troubleshooting. In addition, the existing GSS-API specifications do not have provisions for gracefully dealing with potentially conflicting minor status codes in multi-mechanism implementations, particularly ones that allow for runtime loading of GSS-API mechanisms.

Concrete approaches for improving GSS-API error reporting appear to be somewhat lacking, apart from the "PGSSAPI" proposal by Nico Williams, which adds semantics to the actual pointer value passed as the minor_context argument. Several working group participants find the "PGSSAPI" approach distasteful.

4. Security strength reporting

There is some interest in adding an interface to report the security strength of the established context, for use with implementations of protocols such as SASL. Some debate has taken place about whether a numeric report of security strength is an appropriate means of communicating this information to an application.

5. Programmer friendliness

In the GSS-API C bindings, the gss_accept_sec_context function takes 11 parameters, and the gss_init_sec_context function takes 13. Many of these parameters accept a default value, and in fact application developers sometimes unnecessarily provide non-default values, which often unintentionally results in reduced functionality.

Some programmers find that needing to explicitly loop over gss_init_sec_context and gss_accept_sec_context (as is currently required by a conforming GSS-API application during context establishment) is cumbersome. It may be beneficial to define a simpler interface for programmers who do not require the additional control afforded by explicitly calling the context establishment functions in a loop.

6. Security Considerations

Addition of an interface to report security strength of a GSS-API context enables applications to make better-informed decisions about security policy.

Author's Address

Tom Yu
MIT Kerberos Consortium
77 Massachusetts Ave
Building W92 Room 145
Cambridge, MA 02139
US

Phone: +1 617 253 1753
Email: tlyu@mit.edu