            A Pseudo-Random Function (PRF) API Extension for the
      Generic Security Service Application Program Interface (GSS-API)

Status of This Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements.  Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol.  Distribution of this memo is unlimited.

Copyright Notice

Abstract

   This document defines a Pseudo-Random Function (PRF) extension to the
   Generic Security Service Application Program Interface (GSS-API) for
   keying application protocols given an established GSS-API security
   context.  The primary intended use of this function is to key secure
   session layers that do not or cannot use GSS-API per-message message
   integrity check (MIC) and wrap tokens for session protection.

Table of Contents

1.  Introduction

   A need has arisen for users of the GSS-API to key applications'
   cryptographic protocols using established GSS-API security contexts.
   Such applications can use the GSS-API [RFC2743] for authentication,
   but not for transport security (for whatever reasons), and since the
   GSS-API does not provide a method for obtaining keying material from
   established security contexts, such applications cannot make
   effective use of the GSS-API.

   To address this need, we define a pseudo-random function (PRF)
   extension to the GSS-API.

   Though this document specifies an abstract API as an extension to the
   GSS-API version 2, update 1, and though it specifies the bindings of
   this extension for the C programming language, it does not specify a
   revision of the GSS-API and so does not address the matter of how
   portable applications detect support for and ensure access to this
   extension.  We defer this matter to an expected, comprehensive update
   to the GSS-API.

1.1.  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.  GSS_Pseudo_random()

   Inputs:

   o   context CONTEXT handle,

   o   prf_key INTEGER,

   o   prf_in OCTET STRING,

   o   desired_output_len INTEGER


   Outputs:

   o   major_status INTEGER,

   o   minor_status INTEGER,

   o   prf_out OCTET STRING

Return major_status codes:

o  GSS_S_COMPLETE indicates no error.

o  GSS_S_NO_CONTEXT indicates that a null context has been provided
   as input.

o  GSS_S_CONTEXT_EXPIRED indicates that an expired context has been
   provided as input.

o  GSS_S_UNAVAILABLE indicates that the mechanism lacks support for
   this function or, if the security context is not fully
   established, that the context is not ready to compute the PRF with
   the given prf_key, or that the given prf_key is not available.

o  GSS_S_FAILURE indicates general failure, possibly due to the given
   input data being too large or of zero length, or due to the
   desired_output_len being zero; the minor status code may provide
   additional information.

This function applies the established context's mechanism's keyed
pseudo-random function (PRF) to the input data ('prf_in'), keyed with
key material associated with the given security context and
identified by 'prf_key', and outputs the resulting octet string
('prf_out') of desired_output_len length.

The minimum input data length is one octet.

Mechanisms MUST be able to consume all the provided prf_in input data
that is $2^{14}$ or fewer octets.

If a mechanism cannot consume as much input data as provided by the
caller, then GSS_Pseudo_random() MUST return GSS_S_FAILURE.

The minimum desired_output_len is one.

Mechanisms MUST be able to output at least up to $2^{14}$ octets.

If the implementation cannot produce the desired output due to lack
of resources, then it MUST return GSS_S_FAILURE and MUST set a
suitable minor status code.

The prf_key can take on the following values: GSS_C_PRF_KEY_FULL,
GSS_C_PRF_KEY_PARTIAL, or mechanism-specific values, if any.  This
parameter is intended to distinguish between the best cryptographic
keys that may be available only after full security context
establishment and keys that may be available prior to full security
context establishment.  For some mechanisms, or contexts, those two

prf_key values MAY refer to the same cryptographic keys; for
mechanisms like the Kerberos V GSS-API mechanism [RFC1964] where one
peer may assert a key that may be considered better than the others
they MAY be different keys.

GSS_C_PRF_KEY_PARTIAL corresponds to a key that would have been used
while the security context was partially established, even if it is
fully established when GSS_Pseudo_random() is actually called.
Mechanism-specific prf_key values are intended to refer to any other
keys that may be available.

The GSS_C_PRF_KEY_FULL value corresponds to the best key available
for fully-established security contexts.

GSS_Pseudo_random() has the following properties:

o   its output string MUST be a pseudo-random function [GGM1] [GGM2]
    of the input keyed with key material from the given security
    context -- the chances of getting the same output given different
    input parameters should be exponentially small.

o   when successfully applied to the same inputs by an initiator and
    acceptor using the same security context, it MUST produce the
    _same results_ for both, the initiator and acceptor, even if
    called multiple times (as long as the security context is not
    expired).

o   upon full establishment of a security context, all cryptographic
    keys and/or negotiations used for computing the PRF with any
    prf_key MUST be authenticated (mutually, if mutual authentication
    is in effect for the given security context).

o   the outputs of the mechanism's GSS_Pseudo_random() (for different
    inputs) and its per-message tokens for the given security context
    MUST be "cryptographically separate"; in other words, it must not
    be feasible to recover key material for one mechanism operation or
    transform its tokens and PRF outputs from one to the other given
    only said tokens and PRF outputs.  (This is a fancy way of saying
    that key derivation and strong cryptographic operations and
    constructions must be used.)

o   as implied by the above requirement, it MUST NOT be possible to
    access any raw keys of a security context through
    GSS_Pseudo_random(), no matter what inputs are given.

2.1.  C-Bindings

```
#define GSS_C_PRF_KEY_FULL 0
#define GSS_C_PRF_KEY_PARTIAL 1

OM_uint32 gss_pseudo_random(
  OM_uint32                     *minor_status,
  gss_ctx_id_t                  context,
  int                           prf_key,
  const gss_buffer_t            prf_in,
  ssize_t                       desired_output_len,
  gss_buffer_t                  prf_out
);
```

   Additional major status codes for the C-bindings:

   o  GSS_S_CALL_INACCESSIBLE_READ

   o  GSS_S_CALL_INACCESSIBLE_WRITE

   See [RFC2744].

3.  IANA Considerations

   This document has no IANA considerations currently.  If and when a
   relevant IANA registry of GSS-API symbols is created, then the
   generic and language-specific function names, constant names, and
   constant values described above should be added to such a registry.

4.  Security Considerations

   Care should be taken in properly designing a mechanism's PRF
   function.

   GSS mechanisms' PRF functions should use a key derived from contexts'
   authenticated session keys and should preserve the forward security
   properties of the mechanisms' key exchanges.

   Some mechanisms may support the GSS PRF function with security
   contexts that are not fully established, but applications MUST assume
   that authentication, mutual or otherwise, has not completed until the
   security context is fully established.

   Callers of GSS_Pseudo_random() should avoid accidentally calling it
   with the same inputs.  One useful technique is to prepend to the
   prf_in input string, by convention, a string indicating the intended
   purpose of the PRF output in such a way that unique contexts in which
   the function is called yield unique inputs to it.

Pseudo-random functions are, by their nature, capable of producing
only limited amounts of cryptographically secure output.  The exact
amount of output that one can safely use, unfortunately, varies from
one PRF to another (which prevents us from recommending specific
numbers).  Because of this, we recommend that unless you really know
what you are doing (i.e., you are a cryptographer and are qualified
to pass judgement on cryptographic functions in areas of period,
presence of short cycles, etc.), you limit the amount of the PRF
output used to the necessary minimum.  See [RFC4086] for more
information about "Randomness Requirements for Security".

For some mechanisms, the computational cost of computing
GSS_Pseudo_random() may increase significantly as the length of the
prf_in data and/or the desired_output_length increase.  This means
that if an application can be tricked into providing very large input
octet strings and requesting very long output octet strings, then
that may constitute a denial of service attack on the application;
therefore, applications SHOULD place appropriate limits on the size
of any input octet strings received from their peers without
integrity protection.

5.  References

5.1.  Normative References

   [GGM1]      Goldreich, O., Goldwasser, S., and S. Micali, "How to
               Construct Random Functions", Journal of the ACM, October
               1986.

   [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.

5.2.  Informative References

   [GGM2]      Goldreich, O., Goldwasser, S., and S. Micali, "On the
               Cryptographic Applications of Random Functions",
               Proceedings of CRYPTO 84 on Advances in cryptology, 1985.

   [RFC4086]   Eastlake, D., 3rd, Schiller, J., and S. Crocker,
               "Randomness Requirements for Security", BCP 106, RFC 4086,
               June 2005.

   [RFC1964]   Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC
               1964, June 1996.

Author's Address

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

   EMail: Nicolas.Williams@sun.com

Full Copyright Statement

Intellectual Property

Acknowledgement