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

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

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