Network Working Group
Request for Comments: 4752
Obsoletes: 2222
Category: Standards Track

A. Melnikov, Ed.
Isode

November 2006

The Kerberos V5 ("GSSAPI") Simple Authentication and Security Layer (SASL) Mechanism

Status of This Memo

Copyright Notice

Abstract

The Simple Authentication and Security Layer (SASL) is a framework for adding authentication support to connection-based protocols. This document describes the method for using the Generic Security Service Application Program Interface (GSS-API) Kerberos V5 in the SASL.

This document replaces Section 7.2 of RFC 2222, the definition of the "GSSAPI" SASL mechanism. This document, together with RFC 4422, obsoletes RFC 2222.

Table of Contents

1. Introduction

This specification documents currently deployed Simple Authentication and Security Layer (SASL [SASL]) mechanism supporting the Kerberos V5 [KERBEROS] Generic Security Service Application Program Interface ([GSS-API]) mechanism [RFC4121]. The authentication sequence is described in Section 3. Note that the described authentication sequence has known limitations, in particular, it lacks channel bindings and the number of round-trips required to complete authentication exchange is not minimal. SASL WG is working on a separate document that should address these limitations.

1.1. Relationship to Other Documents

This document, together with RFC 4422, obsoletes RFC 2222 in its entirety. This document replaces Section 7.2 of RFC 2222. The remainder is obsoleted as detailed in Section 1.2 of RFC 4422.

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" in this document are to be interpreted as defined in "Key words for use in RFCs to Indicate Requirement Levels" [KEYWORDS].

3. Kerberos V5 GSS-API Mechanism

The SASL mechanism name for the Kerberos V5 GSS-API mechanism [RFC4121] is "GSSAPI". Though known as the SASL GSSAPI mechanism, the mechanism is specifically tied to Kerberos V5 and GSS-API's Kerberos V5 mechanism.

The GSSAPI SASL mechanism is a "client goes first" SASL mechanism; i.e., it starts with the client sending a "response" created as described in the following section.

The implementation MAY set any GSS-API flags or arguments not mentioned in this specification as is necessary for the implementation to enforce its security policy.

Note that major status codes returned by GSS_Init_sec_context() or GSS_Accept_sec_context() other than GSS_S_COMPLETE or GSS_S_CONTINUE_NEEDED cause authentication failure. Major status codes returned by GSS_Unwrap() other than GSS_S_COMPLETE (without any additional supplementary status codes) cause authentication and/or security layer failure.

3.1. Client Side of Authentication Protocol Exchange

The client calls GSS_Init_sec_context, passing in input_context_handle of 0 (initially), mech_type of the Kerberos V5 GSS-API mechanism [KRB5GSS], chan_binding of NULL, and targ_name equal to output_name from GSS_Import_Name called with input_name_type of GSS_C_NT_HOSTBASED_SERVICE (*) and input_name_string of "service@hostname" where "service" is the service name specified in the protocol's profile, and "hostname" is the fully qualified host name of the server. When calling the GSS_Init_sec_context, the client MUST pass the integ_req_flag of TRUE (**). If the client will be requesting a security layer, it MUST also supply to the GSS_Init_sec_context a mutual_req_flag of TRUE, and a sequence_req_flag of TRUE. If the client will be requesting a security layer providing confidentiality protection, it MUST also supply to the GSS_Init_sec_context a conf_req_flag of TRUE. The client then responds with the resulting output_token. If GSS_Init_sec_context returns GSS_S_CONTINUE_NEEDED, then the client should expect the server to issue a token in a subsequent challenge. The client must pass the token to another call to GSS_Init_sec_context, repeating the actions in this paragraph.

(*) Clients MAY use name types other than GSS_C_NT_HOSTBASED_SERVICE to import servers' acceptor names, but only when they have a priori knowledge that the servers support alternate name types. Otherwise clients MUST use GSS_C_NT_HOSTBASED_SERVICE for importing acceptor names.

(**) Note that RFC 2222 [RFC2222] implementations will not work with GSS-API implementations that require integ_req_flag to be true. No implementations of RFC 1964 [KRB5GSS] or RFC 4121 [RFC4121] that require integ_req_flag to be true are believed to exist and it is expected that any future update to [RFC4121] will require that

integrity be available even in not explicitly requested by the application.

When GSS_Init_sec_context returns GSS_S_COMPLETE, the client examines the context to ensure that it provides a level of protection permitted by the client's security policy. In particular, if the integ_avail flag is not set in the context, then no security layer can be offered or accepted.

If the conf_avail flag is not set in the context, then no security layer with confidentiality can be offered or accepted. If the context is acceptable, the client takes the following actions: If the last call to GSS_Init_sec_context returned an output_token, then the client responds with the output_token, otherwise the client responds with no data. The client should then expect the server to issue a token in a subsequent challenge. The client passes this token to GSS_Unwrap and interprets the first octet of resulting cleartext as a bit-mask specifying the security layers supported by the server and the second through fourth octets as the maximum size output_message the server is able to receive (in network byte order). If the resulting cleartext is not 4 octets long, the client fails the negotiation. The client verifies that the server maximum buffer is 0 if the server does not advertise support for any security layer.

The client then constructs data, with the first octet containing the bit-mask specifying the selected security layer, the second through fourth octets containing in network byte order the maximum size output_message the client is able to receive (which MUST be 0 if the client does not support any security layer), and the remaining octets containing the UTF-8 [UTF8] encoded authorization identity. (Implementation note: The authorization identity is not terminated with the zero-valued (%x00) octet (e.g., the UTF-8 encoding of the NUL (U+0000) character)). The client passes the data to GSS_Wrap with conf_flag set to FALSE and responds with the generated output_message. The client can then consider the server authenticated.

3.2. Server Side of Authentication Protocol Exchange

A server MUST NOT advertise support for the "GSSAPI" SASL mechanism described in this document unless it has acceptor credential for the Kerberos V GSS-API mechanism [KRB5GSS].

The server passes the initial client response to GSS_Accept_sec_context as input_token, setting input_context_handle to 0 (initially), chan_binding of NULL, and a suitable acceptor_cred_handle (see below). If GSS_Accept_sec_context returns GSS_S_CONTINUE_NEEDED, the server returns the generated output_token

to the client in challenge and passes the resulting response to another call to GSS_Accept_sec_context, repeating the actions in this paragraph.

Servers SHOULD use a credential obtained by calling GSS_Acquire_cred or GSS_Add_cred for the GSS_C_NO_NAME desired_name and the Object Identifier (OID) of the Kerberos V5 GSS-API mechanism [KRB5GSS](*). Servers MAY use GSS_C_NO_CREDENTIAL as an acceptor credential handle. Servers MAY use a credential obtained by calling GSS_Acquire_cred or GSS_Add_cred for the server's principal name(s) (**) and the Kerberos V5 GSS-API mechanism [KRB5GSS].

(*) Unlike GSS_Add_cred the GSS_Acquire_cred uses an OID set of GSS-API mechanism as an input parameter. The OID set can be created by using GSS_Create_empty_OID_set and GSS_Add_OID_set_member. It can be freed by calling the GSS_Release_oid_set.

(**) Use of server's principal names having GSS_C_NT_HOSTBASED_SERVICE name type and "service@hostname" format, where "service" is the service name specified in the protocol's profile, and "hostname" is the fully qualified host name of the server, is RECOMMENDED. The server name is generated by calling GSS_Import_name with input_name_type of GSS_C_NT_HOSTBASED_SERVICE and input_name_string of "service@hostname".

Upon successful establishment of the security context (i.e., GSS_Accept_sec_context returns GSS_S_COMPLETE), the server SHOULD verify that the negotiated GSS-API mechanism is indeed Kerberos V5 [KRB5GSS]. This is done by examining the value of the mech_type parameter returned from the GSS_Accept_sec_context call. If the value differs, SASL authentication MUST be aborted.

Upon successful establishment of the security context and if the server used GSS_C_NO_NAME/GSS_C_NO_CREDENTIAL to create acceptor credential handle, the server SHOULD also check using the GSS_Inquire_context that the target_name used by the client matches either

- the GSS_C_NT_HOSTBASED_SERVICE "service@hostname" name syntax, where "service" is the service name specified in the application protocol's profile,

  or

- the GSS_KRB5_NT_PRINCIPAL_NAME [KRB5GSS] name syntax for a two-component principal where the first component matches the service name specified in the application protocol's profile.

When GSS_Accept_sec_context returns GSS_S_COMPLETE, the server examines the context to ensure that it provides a level of protection permitted by the server's security policy. In particular, if the integ_avail flag is not set in the context, then no security layer can be offered or accepted. If the conf_avail flag is not set in the context, then no security layer with confidentiality can be offered or accepted.

If the context is acceptable, the server takes the following actions: If the last call to GSS_Accept_sec_context returned an output_token, the server returns it to the client in a challenge and expects a reply from the client with no data. Whether or not an output_token was returned (and after receipt of any response from the client to such an output_token), the server then constructs 4 octets of data, with the first octet containing a bit-mask specifying the security layers supported by the server and the second through fourth octets containing in network byte order the maximum size output_token the server is able to receive (which MUST be 0 if the server does not support any security layer). The server must then pass the plaintext to GSS_Wrap with conf_flag set to FALSE and issue the generated output_message to the client in a challenge.

The server must then pass the resulting response to GSS_Unwrap and interpret the first octet of resulting cleartext as the bit-mask for the selected security layer, the second through fourth octets as the maximum size output_message the client is able to receive (in network byte order), and the remaining octets as the authorization identity. The server verifies that the client has selected a security layer that was offered and that the client maximum buffer is 0 if no security layer was chosen. The server must verify that the src_name is authorized to act as the authorization identity. After these verifications, the authentication process is complete. The server is not expected to return any additional data with the success indicator.

3.3. Security Layer

The security layers and their corresponding bit-masks are as follows:

```
1 No security layer
2 Integrity protection.
  Sender calls GSS_Wrap with conf_flag set to FALSE
4 Confidentiality protection.
  Sender calls GSS_Wrap with conf_flag set to TRUE
```

Other bit-masks may be defined in the future; bits that are not understood must be negotiated off.

When decoding any received data with GSS_Unwrap, the major_status other than the GSS_S_COMPLETE MUST be treated as a fatal error.

Note that SASL negotiates the maximum size of the output_message to send. Implementations can use the GSS_Wrap_size_limit call to determine the corresponding maximum size input_message.

4. IANA Considerations

IANA modified the existing registration for "GSSAPI" as follows:

Family of SASL mechanisms: NO

SASL mechanism name: GSSAPI

Security considerations: See Section 5 of RFC 4752

Published specification: RFC 4752

Person & email address to contact for further information:
Alexey Melnikov <Alexey.Melnikov@isode.com>

Intended usage: COMMON

Owner/Change controller: iesg@ietf.org

Additional information: This mechanism is for the Kerberos V5 mechanism of GSS-API.

5. Security Considerations

Security issues are discussed throughout this memo.

When constructing the input_name_string, the client SHOULD NOT canonicalize the server's fully qualified domain name using an insecure or untrusted directory service.

For compatibility with deployed software, this document requires that the chan_binding (channel bindings) parameter to GSS_Init_sec_context and GSS_Accept_sec_context be NULL, hence disallowing use of GSS-API support for channel bindings. GSS-API channel bindings in SASL is expected to be supported via a new GSS-API family of SASL mechanisms (to be introduced in a future document).

Additional security considerations are in the [SASL] and [GSS-API] specifications. Additional security considerations for the GSS-API mechanism can be found in [KRB5GSS] and [KERBEROS].

6. Acknowledgements

This document replaces Section 7.2 of RFC 2222 [RFC2222] by John G. Myers. He also contributed significantly to this revision.

Lawrence Greenfield converted text of this document to the XML format.

Contributions of many members of the SASL mailing list are gratefully acknowledged, in particular comments from Chris Newman, Nicolas Williams, Jeffrey Hutzelman, Sam Hartman, Mark Crispin, and Martin Rex.

7. Changes since RFC 2222

RFC 2078 [RFC2078] specifies the version of GSS-API used by RFC 2222 [RFC2222], which provided the original version of this specification. That version of GSS-API did not provide the integ_integ_avail flag as an input to GSS_Init_sec_context. Instead, integrity was always requested. RFC 4422 [SASL] requires that when possible, the security layer negotiation be integrity protected. To meet this requirement and as part of moving from RFC 2078 [RFC2078] to RFC 2743 [GSS-API], this specification requires that clients request integrity from GSS_Init_sec_context so they can use GSS_Wrap to protect the security layer negotiation. This specification does not require that the mechanism offer the integrity security layer, simply that the security layer negotiation be wrapped.

8. References

8.1. Normative References

[GSS-API] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000.

[KERBEROS] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[KRB5GSS] Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, June 1996.

[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, July 2005.

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

[UTF8] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

8.2. Informative References

[RFC2078] Linn, J., "Generic Security Service Application Program Interface, Version 2", RFC 2078, January 1997.

[RFC2222] Myers, J., "Simple Authentication and Security Layer (SASL)", RFC 2222, October 1997.

Editor'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/

Full Copyright Statement

Intellectual Property

Acknowledgement

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