NETWORK WORKING GROUP
Internet-Draft
Obsoletes: 2478 (if approved)
Expires: May 23, 2005

L. Zhu
P. Leach
K. Jaganathan
Microsoft Corporation
W. Ingersoll
Sun Microsystems

November 22, 2004

The Simple and Protected GSS-API Negotiation Mechanism
draft-ietf-kitten-2478bis-00

Status of this Memo

Copyright Notice

Abstract

This document specifies a negotiation mechanism for the Generic Security Service Application Program Interface (GSS-API) which is described in RFC 2743.

GSS-API peers can use this negotiation mechanism to choose from a common set of security mechanisms.

Table of Contents

1. Introduction

The GSS-API [RFC2743] provides a generic interface which can be layered atop different security mechanisms such that if communicating peers acquire GSS-API credentials for the same security mechanism, then a security context may be established between them (subject to policy). However, GSS-API doesn't prescribe the method by which GSS-API peers can establish whether they have a common security mechanism.

The Simple and Protected GSS-API Negotiation (SPNEGO) mechanism defined here is a pseudo security mechanism, represented by the Object Identifier iso.org.dod.internet.security.mechanism.snego (1.3.6.1.5.5.2), which enables GSS-API peers to determine in-band whether their credentials share common GSS-API security mechanism(s), and if so, to invoke normal security context establishment for a selected common security mechanism. This is most useful for applications that are based on GSS-API implementations and multiple mechanisms are shared between the peers.

The SPNEGO mechanism negotiation is based on the following negotiation model: the initiator proposes a list of security mechanism(s), in its preference order (favorite choice first), the acceptor (also known as the target) either accepts the initiator's preferred security mechanism (the first in the list), or chooses one that is available from the offered list, or rejects the proposed value(s). The target then informs the initiator of its choice.

Once a common security mechanism is chosen, it MAY also negotiate mechanism-specific options during its context establishment, but that will be inside the mechanism tokens and invisible to this protocol.

If per-message integrity services are available on the established mechanism security context, the peers can then exchange MIC tokens to ensure that the mechanism list was not tampered with. This MIC token exchange is OPTIONAL if no interference could have material impact on the negotiation, i.e., when the selected mechanism is the first choice for both peers.

In order to avoid an extra round trip, the first security token of the preferred mechanism SHOULD be embedded in the initial negotiation message (as defined in Section 4.2). This mechanism token is referred to as the optimistic token in this document. If the selected mechanism matches the initiator's preferred mechanism, no additional round trips need to be incurred by using this protocol. In addition, by using the optimistic token, the initiator can recover from a non-fatal error in producing the first token before a mechanism can be selected. Implementations, however, MAY omit the

optimistic token, to avoid the cost of generating it in cases where the initiator's preferred mechanism is not selected by the acceptor.

SPNEGO uses the concepts developed in the GSS-API specification [RFC2743]. The negotiation data is encapsulated in context-level tokens. Therefore, callers of the GSS-API do not need to be aware of the existence of the negotiation tokens but only of the new pseudo-security mechanism. A failure in the negotiation phase causes a major status code to be returned: GSS_S_BAD_MECH.

2. 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].

3. Negotiation Protocol

When the established mechanism context provides for integrity protection, the mechanism negotiation can be protected. When acquiring negotiated security mechanism tokens, per-message integrity services are always requested by the SPNEGO mechanism.

When the established mechanism context supports per-message integrity services, SPNEGO guarantees that the selected mechanism is mutually preferred.

This section describes the negotiation process of this protocol.

3.1 Negotiation Description

The first negotiation token sent by the initiator contains an ordered list of mechanisms (in preference order, favorite choice first), and optionally the initial security token for the preferred mechanism of the initiator (i.e., the first in the list). The list of security mechanisms available for negotiation is based on the credentials being used.

The target then processes the token from the initiator. This will result in one of four possible states (as defined in Section 4.2.2): accept_completed, accept_incomplete, reject, or request_mic. A reject state will terminate the negotiation; an accept_completed state indicates that not only was the initiator-selected mechanism acceptable to the target, but that the initial token was sufficient to complete the authentication; an accept_incomplete state indicates that further message exchange is needed but the MIC token exchange as described in Section 5 is OPITONAL; a request_mic state (this state can only be present in the first reply message from the target) indicates the MIC token exchange is REQUIRED if per-message integrity services are available.

Unless the preference order is specified by the application (see Appendix A), the policy by which the target chooses a mechanism is an implementation-specific local matter. In the absence of application specified preference order or other policy, the target SHALL choose the first mechanism in the initiator proposed list for which it has valid credentials.

In case of a successful negotiation, the security mechanism in the first reply message represents the value suitable for the target, and picked up from the list offered by the initiator. A context level token for a reject state is OPTIONAL.

Once a mechanism has been selected, the tokens specific to the

selected mechanism are carried within the negotiation tokens.

Lastly, MIC tokens MAY be exchanged to ensure the authenticity of the mechanism list as seen by the target.

To avoid conflicts with the use of MIC tokens by SPNEGO, partially-established contexts are not used for per-message calls: the prot_ready_state [RFC2743] will be false even if the underlying mechanism would return true natively.

3.2 Negotiation Procedure

The basic form of the procedure assumes that per-message integrity services are available on the established mechanism context, and it is summarized as follows:

(a) The GSS-API initiator invokes GSS_Init_sec_context() as normal, but requests (either explicitly, with the negotiation mechanism, or through accepting a default, when the default is this negotiation mechanism) that SPNEGO is used.

(b) The initiator GSS-API implementation emits a negotiation token containing a list of supported security mechanisms (possible just one mechanism) for the credentials used for this context establishment, and optionally an initial security token for the first mechanism from that list.

(c) The GSS-API initiator application sends the token to the target application. The GSS-API target application deposits the token through invoking GSS_Accept_sec_context(). The acceptor will do one of the following:

  (I) No proposed mechanism is acceptable, the negotiation SHALL be terminated. GSS_Accept_sec_context indicates GSS_S_BAD_MECH. The acceptor MAY output a negotiation token containing a reject state.

  (II) If either the initiator's preferred mechanism is not accepted by the target, or this mechanism is accepted but it is not the most preferred mechanism available for the acceptor (see Section 3.1 and Section 5), GSS_Accept_sec_context() indicates GSS_S_CONTINUE_NEEDED. The acceptor MUST output a negotiation token containing a request_mic state.

  (III) Otherwise, GSS_Accept_sec_conext() indicates GSS_S_COMPLETE or GSS_S_CONTINUE_NEEDED, depending on if at least one additional negotiation token from the initiator is needed to establish this context. The acceptor outputs a negotiation

token containing an accept_complete or accept_incomplete state, respectively.

If the initiator's preferred mechanism is accepted, and an optimistic mechanism token was included, this mechanism token MUST be deposited to the selected mechanism through invoking GSS_Accept_sec_context() and if a response mechanism token is emitted, it MUST be included in the response negotiation token. Otherwise, the target will not emit a response mechanism token in the first reply.

(d) The GSS-API target application returns the negotiation token to the initiator application. The GSS-API initiator application deposits the token through invoking GSS_Init_sec_context(). The security context initialization is then continued according to the standard GSS-API conventions for the selected mechanism, where the tokens of the selected mechanism are encapsulated until the GSS_S_COMPLETE is returned for both the initiator and the target by the selected security mechanism.

(e) MIC tokens are then either skipped or exchanged according to Section 5.

Note that the *_req_flag input parameters for context establishment are relative to the selected mechanism, as are the *_state output parameters. i.e., these parameters are not applicable to the negotiation process per se.

On receipt of a negotiation token on the target side, a GSS-API implementation that does not support negotiation would indicate the GSS_S_BAD_MECH status as if a particular basic security mechanism had been requested but was not supported.

When GSS_Acquire_cred is invoked with this SPNEGO mechanism as desired_mechs, an implementation-specific default credential is used to carry on the negotiation. A set of mechanisms as specified locally by the system administrator is then available for negotiation. If there is a desire for the caller to make its own choice, then an additional API has to be used (see Appendix A).

4. Token Definitions

The type definitions in this section assume an ASN.1 module definition of the following form:

```
SPNEGOASNOneSpec {
    iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanism(5) snego (2) modules(4) spec2(2)
} DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- rest of definitions here

END
```

This specifies that the tagging context for the module will be explicit and non-automatic.

The encoding of SPNEGO protocol messages shall obey the Distinguished Encoding Rules (DER) of ASN.1 as described in [X690].

4.1 Mechanism Types

In this negotiation model, each OID represents one GSS-API mechanism or one variant of it according to [RFC2743].

```
MechType ::= OBJECT IDENTIFIER
    -- OID represents each security mechanism as suggested by
    -- [RFC2743]

MechTypeList ::= SEQUENCE OF MechType
```

4.2 Negotiation Tokens

The syntax of the initial negotiation tokens follows the initialContextToken syntax defined in Section 3.1 of [RFC2743]. The SPNEGO pseudo mechanism is identified by the Object Identifier specified in Section 1. Subsequent tokens are not encapsulated in this GSS-API generic token framing.

This section specifies the syntax of the inner token for the initial message, and the syntax of subsequent context establishment tokens.

```
NegotiationToken ::= CHOICE {
    negTokenInit    [0] NegTokenInit,
```

```
            negTokenResp    [1] negTokenResp
        }
```

4.2.1 negTokenInit

```
        NegTokenInit ::= SEQUENCE {
            mechTypes       [0] MechTypeList,
            reqFlags        [1] ContextFlags  OPTIONAL,
            mechToken       [2] OCTET STRING  OPTIONAL,
            mechListMIC     [3] OCTET STRING  OPTIONAL,
            ...
        }
        ContextFlags ::= BIT STRING {
            delegFlag       (0),
            mutualFlag      (1),
            replayFlag      (2),
            sequenceFlag    (3),
            anonFlag        (4),
            confFlag        (5),
            integFlag       (6)
        }
```

This is the syntax for the inner token of the initial negotiation message.

mechTypes

This field contains one or more security mechanisms available for the initiator in preference order (favorite choice first).

reqFlags

This field, if present, contains the service options that are requested to establish the context. The context flags SHOULD be filled in from the req_flags parameter of GSS_Init_sec_context(). This field SHALL NOT have impact on the negotiation.

mechToken

This field, is present, contains the optimistic security mechanism token.

mechlistMIC

This field, is present, contains a MIC token, which is computed according to Section 5, for the mechanism list in the initial negotiation message.

4.2.2 negTokenResp

```
NegTokenResp ::= SEQUENCE {
    negResult       [0] ENUMERATED {
        accept_completed    (0),
        accept_incomplete   (1),
        reject              (2),
        request_mic         (3)
    },
    supportedMech   [1] MechType      OPTIONAL,
    responseToken   [2] OCTET STRING  OPTIONAL,
    mechListMIC     [3] OCTET STRING  OPTIONAL,
    ...
}
```

This is the syntax for all subsequent negotiation messages.

negResult

This field contains the state of the negotiation. This can be:

accept_completed
No further negotiation message from the peer is expected, and the security context is established for the sender.

accept_incomplete
At least one more negotiation message from the peer is needed to establish the security context.

reject
The sender terminates the negotiation.

request_mic
The sender indicates that the exchange of MIC tokens, as described in Section 5, will be REQUIRED if per-message integrity services are available on the mechanism context to be established. This value SHALL only be present in the first reply from the target.

supportedMech

This field SHALL only be present in the first reply from the target. It is a choice from the mechanism(s) offered by the initiator.

ResponseToken

The field, if present, contains tokens specific to the mechanism selected.

mechlistMIC

This field, is present, contains a MIC token, which is computed according to Section 5, for the mechanism list in the initial negotiation message.

5. Processing of mechListMIC

If the mechanism selected by the negotiation does not support integrity protection, then no mechlistMIC token is used. Otherwise if the initiator's preferred mechanism is accepted and it is also the most preferred mechanism available for the acceptor (there is no mechanism which, had it been present in the mechanism list, the acceptor would have preferred over the accepted mechanism), then the MIC token exchange, as described later in this section, is OPTIONAL. In all other cases, MIC tokens MUST be exchanged after the mechanism context is fully established.

It is assumed that per-message integrity services are available on the established mechanism context in the following procedure for processing MIC tokens of the initiator's mechanism list.

a) The mechlistMIC token (or simply the MIC token) is computed through invoking GSS_GetMIC(): the input context_handle is the established mechanism context, the input qop_req is 0, and the input message is the mechTypes field in the initial negotiation message (only the "value" portion, omitting the tag and length, of the ASN.1 encoding for that field is included).

b) If the selected mechanism uses an even number of mechanism tokens (namely the acceptor sends the last mechanism token), the acceptor does the following when emitting the negotiation message containing the last mechanism token: if the MIC token exchange is not required, GSS_Accept_sec_context() either indicates GSS_S_COMPLETE and does not include a mechlistMIC token, or indicates GSS_S_CONTINUE_NEEDED and includes a mechlistMIC token and an accept_incomplete state; if the MIC token exchange is required, GSS_Accept_sec_context() indicates GSS_S_CONTINUE_NEEDED, and includes a mechlistMIC token. Acceptors who wish to be compatible with legacy Windows SPNEGO implementations as described in Appendix B shall not generate a mechlistMIC token when the MIC token exchange is not required. The initiator then processes the last mechanism token, and does one of the following:

   (I) If a mechlistMIC token was included, and is correctly verified, GSS_Init_sec_context() indicates GSS_S_COMPLETE. The output negotiation message contains a mechlistMIC token, and an accept_complete state. The acceptor MUST then verify this mechlistMIC token.

(II) If a mechlistMIC token was included but is incorrect, the negotiation SHALL be terminated. GSS_Accept_sec_context() indicates GSS_S_DEFECTIVE_TOKEN.

(III) If no mechlistMIC token was included, and the MIC token exchange is not required, GSS_Init_sec_context() indicates GSS_S_COMPLETE with no output token.

(IV) If no mechlistMIC token was included, but the MIC token exchange is required, the negotiation SHALL be terminated. GSS_Accept_sec_context() indicates GSS_S_DEFECTIVE_TOKEN.

c) In the case that the chosen mechanism uses an odd number of mechanism tokens (namely the initiator sends the last mechanism token), the initiator does the following when emitting the negotiation message containing the last mechanism token: if the negResult state was request_mic in the first reply from the target, a mechlistMIC token MUST be included, otherwise the mechlistMIC token is OPTIONAL. GSS_Init_sec_context() indicates GSS_S_CONTINUE_NEEDED. Initiators who wish to be compatible with legacy Windows SPNEGO implementations as described in Appendix B shall not generate a mechlistMIC token when the MIC token exchange is not required. The acceptor then processes the last mechanism token, and does one of the following:

(I) If a mechlistMIC token was included, and is correctly verified, GSS_Accept_sec_context() indicates GSS_S_COMPLETE. The output negotiation message contains a mechlistMIC token, and an accept_complete state. The initiator MUST then verify this mechlistMIC token.

(II) If a mechlistMIC token was included but is incorrect, the negotiation SHALL be terminated. GSS_Accept_sec_context() indicates GSS_S_DEFECTIVE_TOKEN.

(III) If no mechlistMIC token was included and the mechlistMIC token exchange is not required, GSS_Accept_sec_context() indicates GSS_S_COMPLETE. The output negotiation message contains an accept_complete state.

(IV) If no mechlistMIC token was included and the acceptor sent a request_mic state in the first reply message (the exchange of MIC tokens is required), the negotiation SHALL be terminated. GSS_Accept_sec_context() indicates GSS_S_DEFECTIVE_TOKEN.

6. Extensibility

Two mechanisms are provided by extensibility. First, the ASN.1 structures in this specification MAY be expanded by IETF standards action. Implementations receiving unknown fields MUST ignore these fields.

Secondly, OIDs corresponding to a desired mechanism attribute may be included in the set of preferred mechanisms by an initiator. The acceptor can choose to honor this request by preferring mechanisms that have that attribute. Future work within the Kitten working group is expected to standardize common attributes that SPNEGO mechanisms may wish to support. At this time it is sufficient to say that initiators MAY include OIDs that do not correspond to mechanisms but instead correspond to desired mechanism attributes in their requests. Such OIDs MAY influence the acceptor's choice of mechanism. As discussed in Section 5, if there are mechanisms that if present in the initiator's list of mechanisms might be preferred by the acceptor to the initiator's preferred mechanism, the acceptor MUST demand the MIC token exchange. As a consequence, acceptors MUST demand the MIC token exchange if they support negotiation of attributes not available in the initiator's preferred mechanism regardless of whether the initiator actually requested these attributes.

7. Security Considerations

In order to produce the MIC token for the mechanism list, the mechanism must provide integrity protection. When the selected mechanism does not support integrity protection, then the negotiation is vulnerable: an active attacker can force it to use a security mechanism that is not mutually preferred but is acceptable anyway to the target.

When per-message integrity services are available on the established mechanism context, and there was an alteration of the mechanism list by an adversary such that a common mechanism that is not mutually preferred could be selected, this protocol provides the following guarantees: if the last mechanism token is sent by the initiator, both peers shall fail; if the last mechanism token is sent by the acceptor, the acceptor shall not complete and the initiator at worst shall complete with its preferred mechanism being selected. The negotiation may not be terminated if an alteration was made but it had no material impact.

The protection of the negotiation depends on the strength of the integrity protection. In particular, the strength of SPNEGO is no stronger than the integrity protection of the weakest mechanism acceptable to GSS-API peers.

In all cases, the communicating peers are exposed to the denial of service threat.

8. IANA Considerations

This document has no actions for IANA.

9. Acknowledgments

The authors wish to thank Sam Hartman, Nicolas Williams, Ken Raeburn, Jeff Altman, Cristian Ilac and Martin Rex for their comments and suggestions on earlier versions of this document.

Eric Baize and Denis Pinkas wrote the original SPNEGO specification [RFC2478], of which some of the text has been retained in this document.

10 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2478] Baize, E. and D. Pinkas, "The Simple and Protected GSS-API Negotiation Mechanism", RFC 2478, December 1998.

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000.

[X690] ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), ITU-T Recommendation X.690 (1997) | ISO/IEC International Standard 8825-1:1998.

Authors' Addresses

Larry Zhu
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
US

EMail: lzhu@microsoft.com

Paul Leach
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
US

EMail: paulle@microsoft.com

Karthik Jaganathan
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
US

EMail: karthikj@microsoft.com

Wyllys Ingersoll
Sun Microsystems
1775 Wiehle Avenue, 2nd Floor
Reston, VA 20190
US

EMail: wyllys.ingersoll@sun.com

Appendix A. GSS-API Negotiation Support API

In order to provide to a GSS-API caller (either the initiator or the target or both) the ability to choose among the set of supported mechanisms a reduced set of mechanisms for negotiation, two additional APIs are defined:

- o GSS_Get_neg_mechs() indicates the set of security mechanisms available on the local system to the caller for negotiation, based on the credentials being used.
- o GSS_Set_neg_mechs() specifies the set of security mechanisms to be used on the local system by the caller for negotiation, for the given credentials.

A.1 GSS_Set_neg_mechs call

Inputs:

```
o  cred_handle CREDENTIAL HANDLE, -- NULL specifies default
   -- credentials
o  mech_set SET OF OBJECT IDENTIFIER
```

Outputs:

```
o  major_status INTEGER,
o  minor_status INTEGER
```

Return major_status codes:

- o GSS_S_COMPLETE indicates that the set of security mechanisms available for negotiation has been set to mech_set.
- o GSS_S_FAILURE indicates that the requested operation could not be performed for reasons unspecified at the GSS-API level.

Allows callers to specify the set of security mechanisms that may be negotiated with the credential identified by cred_handle. This call is intended for support of specialized callers who need to restrict the set of negotiable security mechanisms from the set of all security mechanisms available to the caller (based on available credentials). Note that if more than one mechanism is specified in mech_set, the order in which those mechanisms are specified implies a relative preference.

A.2 GSS_Get_neg_mechs call

Input:

- o cred_handle CREDENTIAL HANDLE -- NULL specifies default -- credentials

Outputs:

- o major_status INTEGER,
- o minor_status INTEGER,
- o mech_set SET OF OBJECT IDENTIFIER

Return major_status codes:

- o GSS_S_COMPLETE indicates that the set of security mechanisms available for negotiation has been returned in mech_set.
- o GSS_S_FAILURE indicates that the requested operation could not be performed for reasons unspecified at the GSS-API level.

Allows callers to determine the set of security mechanisms available for negotiation with the credential identified by cred_handle. This call is intended for support of specialized callers who need to reduce the set of negotiable security mechanisms from the set of supported security mechanisms available to the caller (based on available credentials).

Note: The GSS_Indicate_mechs() function indicates the full set of mechanism types available on the local system. Since this call has no input parameter, the returned set is not necessarily available for all credentials.

Appendix B. Changes since RFC2478

SPNEGO implementations in Windows 2000/Windows XP/Windows Server 2003 have the following behavior: no mechlistMIC is produced, and mechlistMIC is not processed if one is provided; if the initiator sends the last mechanism token, the acceptor will send back a negotiation token with an accept_complete state and no mechlistMIC token. In addition, the OID (1.2.840.48018.1.2.2) can be used to identify the GSS-API Kerberos Version 5 mechanism.

The following changes have been made to be compatible with these legacy implementations.

* NegTokenTarg is changed to negTokenResp and it is the message format for all subsequent negotiation tokens.
* NegTokenInit is the message for the initial token and that token only.
* mechTypes in negTokenInit is not optional.
* negResult is not optional in the negTokenResp token.
* Two MIC tokens are exchanged, one in each direction.
* If the selected mechanism is also the most preferred mechanism for both peers, it is safe to omit the MIC tokens.

If at least one of the two peers implements the pseudo mechanism in this document, the negotiation is protected.

The following changes are to address the problems in RFC 2478.

* reqFlags is not protected therefore it should not impact the negotiation.
* DER encoding is required.
* GSS_GetMIC() input is clarified.
* Per-message integrity services are requested for the negotiated mechanism.

Acknowledgment

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