           The Simple and Protected GSS-API Negotiation Mechanism

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.

1. ABSTRACT

   This document specifies a Security Negotiation Mechanism for the
   Generic Security Service Application Program Interface (GSS-API)
   which is described in [1].

   The GSS-API 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 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 which support multiple security mechanisms.

   This allows to negotiate different security mechanisms, different
   options within a given security mechanism or different options from
   several security mechanisms.

Once the common security mechanism is identified, the security
mechanism may also negotiate mechanism-specific options during its
context establishment. This will be inside the mechanism tokens, and
invisible to the SPNEGO protocol.

The simple and protected GSS-API mechanism negotiation is based on
the following negotiation model : the initiator proposes one security
mechanism or an ordered list of security mechanisms, the target
either accepts the proposed security mechanism, or chooses one from
an offered set, or rejects the proposed value(s). The target then
informs the initiator of its choice.

In its basic form this protocol requires an extra-round trip. Network
connection setup is a critical performance characteristic of any
network infrastructure and extra round trips over WAN links, packet
radio networks, etc. really make a difference. In order to avoid such
an extra round trip the initial security token of the preferred
mechanism for the initiator may be embedded in the initial token. If
the target preferred mechanism matches the initiator's preferred
mechanism, no additional round trips are incurred by using the
negotiation protocol.

The simple and protected GSS-API mechanism negotiation provides a
technique to protect the negotiation that must be used when the
underlying mechanism selected by the target is capable of integrity
protection.

When all the mechanisms proposed by the initiator support integrity
protection or when the selected mechanism supports integrity
protection, then the negotiation mechanism becomes protected since
this guarantees that the appropriate mechanism supported by both
peers has been selected.

The Simple and Protected GSS-API Negotiation Mechanism uses the
concepts developed in the GSS-API specification [1]. 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. NEGOTIATION MODEL

2.1. Negotiation description

   The model for security mechanism negotiation reuses a subset of the
   concepts specified in [2].

   Each OID represents one GSS-API mechanism or one variant of it.

   - When one security mechanism is proposed by the initiator, it
      represents the only security mechanism supported or selected
      (when the additional APIs defined in the Annex A are used) by the
      initiator.

   - When several security mechanisms are proposed by the initiator,
      they represent a set of security mechanisms supported or selected
      (when the additional APIs defined in the Annex A are used) by the
      initiator.

   The first negotiation token sent by the initiator contains an ordered
   list of mechanisms, a set of options (e.g. deleg, replay, conf flags)
   that should be supported by the selected mechanism and optionally the
   initial security token for the desired mechanism of the initiator
   (i.e. the first of the list).

   The first negotiation token sent by the target contains the result of
   the negotiation (accept_completed, accept_incomplete or reject) and,
   in case of accept, the agreed security mechanism. It may also include
   the response to the initial security token from the initiator, when
   the first proposed mechanism of the initiator has been selected. When
   the first mechanism is acceptable to the target,it should respond to
   the initial security token for the desired mechanism of the initiator
   when it is present. However, if this is not possible, the target can
   simply ignore it and omit the responseToken from the first reply.

   Implementations that can piggyback the initial token will be rewarded
   by faster connection setup.

   In case of a successful negotiation, the security mechanism
   represents the value suitable for the target, and picked up from the
   list offered by the initiator. The policy by which the target
   chooses a mechanism is an implementation-specific local matter. In
   the absence of other policy, the target should chose the first
   mechanism in the list for which valid credentials are available.

   Once a mechanism has been selected, the tokens specific to the
   selected mechanism are carried within the negotiation tokens (in the
   mechToken for the initiator and in the responseToken for the target).

2.2. Negotiation procedure

   The negotiation procedure is summarised 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 the negotiation
       mechanism) that the Simple and Protected GSS-API Negotiation
       Mechanism be used;

   (b) the initiator GSS-API implementation emits a negotiation token
       containing a list of supported security mechanisms for the
       credentials used for this context establishment, and optionally
       an initial security token for the first mechanism from that list
       (i.e. the preferred mechanism), and indicates
       GSS_S_CONTINUE_NEEDED status;

   (c) The GSS-API initiator sends the token to the target application;

   (d) The GSS-API target deposits the token through invoking
       GSS_Accept_sec_context. The target GSS-API implementation emits a
       negotiation token containing which if any of the proposed
       mechanisms it supports (or has selected).

   If the mechanism selected by the target matches the preferred
   mechanism identified by the initiator and the initiator provides a
   mechToken, the negotiation token response may contain also an initial
   security token from that mechanism.

   If the preferred mechanism is accepted, GSS_Accept_sec_context()
   indicates GSS_S_COMPLETE when unilateral or mutual authentication has
   been performed and involves a single token in either direction.

   If a proposed mechanism is accepted, and it was not the preferred
   mechanism, or if the first negotiation token sent by the initiator
   did not included a mechToken, then the negotiation token response
   sent by the target may contain also a response token from that
   mechanism which transmits mechanism-specific information (e.g. to
   transmit a certificate). The initiator may ignore such an initial
   token if it is not prepared to process it.

   If a proposed mechanism other than the preferred mechanism is
   accepted, or the preferred mechanism is accepted but involves
   multiple exchanges (e.g. challenge-response authentication), then
   GSS_Accept_sec_context() indicates GSS_S_CONTINUE_NEEDED status.

If the proposed mechanism(s) are rejected, GSS_Accept_sec_context()
indicates GSS_S_BAD_MECH status. The security context initialisation
has failed.

(e) The GSS-API target returns the token to the initiator
    application;

(f) The GSS-API initiator deposits the token through invoking
    GSS_Init_sec_context.

GSS_Init_sec_context() may then indicate GSS_S_CONTINUE_NEEDED,
GSS_S_COMPLETE or GSS_S_BAD_MECH status.

   The GSS_S_BAD_MECH status is returned when the negotiation token
   carries a reject result or when the negotiation token carries an
   accept result and the mechanism selected by the target is not
   included in the initial list sent by the initiator.

   The GSS_S_BAD_MIC status is returned when the selected mechanism
   supports a MIC token but the MIC computed over the list of
   mechanisms sent by the initiator is missing or incorrect.

   If the negotiation token carries a reject result, the context
   establishment is impossible. For example, a rejection will occur
   if the target doesn't support the initiator's proposed mechanism
   type(s). Upon failure of the mechanism negotiation procedure, the
   mech_type output parameter value is the negotiation mechanism
   type.

   The GSS_S_CONTINUE_NEEDED status is returned when the negotiation
   token carries an accept result and further tokens must be
   transferred in order to complete context establishment for the
   selected mechanism. In that case GSS_Init_sec_context() returns an
   initial context token as output_token (with the selected
   mechanism's context token encapsulated within that output_token).

   The initiator then sends the output_token to the target. The
   security context initialisation 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.
   When GSS_S_CONTINUE_NEEDED is returned, the mech_type output
   parameter is not yet valid.

   When GSS_S_COMPLETE is returned, the mech_type output parameter
   indicates the selected mechanism. When the final negotiation token
   does not contain a MIC, the initiator GSS-API implementation must
   check the returned/selected mechanism is on the originally

submitted list of mechanisms and also verify that the selected
mechanism is not able to support a MIC. When the final negotiation
token contains a MIC over the initial mechanisms list sent by the
initiator, the MIC must be verified.

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.

The initiator GSS-API calling application may need to know when the
negotiation exchanges were protected or not. For this, when
GSS_S_COMPLETE is returned, it can simply test the integ_avail flag.
When this flag is set it indicates that the negotiation was
protected.

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 the negotiation 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).

3. DATA ELEMENTS

3.1. Mechanism Type

   MechType::= OBJECT IDENTIFIER

   mechType
        Each security mechanism is as defined in [1].

3.2. Negotiation Tokens

   The syntax of the negotiation tokens follows the InitialContextToken
   syntax defined in [1]. The security mechanism of the initial
   negotiation token is identified by the Object Identifier
   iso.org.dod.internet.security.mechanism.snego (1.3.6.1.5.5.2).

3.2.1. Syntax

   This section specifies the syntax of the corresponding
   "innerContextToken" field for the first token and subsequent
   negotiation tokens. During the mechanism negociation, the
   "innerContextToken" field contains the ASN.1 structure
   "NegociationToken" given below, encoded using the DER encoding
   conventions.

NegotiationToken ::= CHOICE {
                             negTokenInit [0] NegTokenInit,
                             negTokenTarg [1] NegTokenTarg }

MechTypeList ::= SEQUENCE OF MechType

NegTokenInit ::= SEQUENCE {
                             mechTypes   [0] MechTypeList OPTIONAL,
                             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)
}

negTokenInit
     Negotiation token sent by the initiator to the target, which
     contains, for the first token sent, one or more security mechanisms
     supported by the initiator (as indicated in the field mechTypes)
     and the service options (reqFlags) that are requested to establish
     the context. The context flags should be filled in from the
     req_flags parameter of init_sec_context().

     The mechToken field is optional for the first token sent that all
     target implementations would not have to support. However for those
     targets that do support piggybacking the initial mechToken, an
     optimistic negotiation response is possible. Otherwise the
     mechToken is used to carry the tokens specific to the mechanism
     selected.

The mechListMIC is an optional field. In the case that the chosen
mechanism supports integrity, the initiator may optionally include
a mechListMIC which is the result of a GetMIC of the MechTypes in
the initial NegTokenInit and return GSS_S_COMPLETE.

When the chosen mechanism uses an odd number of messages, the final
mechanism token will be sent from the initiator to the acceptor. In
this case, there is a tradeoff between using the optimal number of
messages, or using an additional message from the acceptor to the
initiator in order to give the initiator assurance that no
modification of the initiator's mechanism list occurred. The
implementation can choose which tradeoff to make (see section 4.2.2
for further details for the processing of that field).

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

negTokenTarg
     Negotiation token returned by the target to the initiator which
     contains, for the first token returned, a global negotiation result
     and the security mechanism selected (if any).

negResult
     The result accept_completed indicates that a context has been
     successfully established, while the result accept_incomplete
     indicates that additional token exchanges are needed.

          Note: For the case where (a) a single-token context setup is
          used and (b) the preferred mechanism does not support the
          integrity facility which would cause a mechListMIC to be
          generated and enclosed, this feature allows to make a
          difference between a mechToken sent by the initiator but not
          processed by the target (accept_incomplete) and a mechToken
          sent by the initiator and processed by the target
          (accept_completed).

     For those targets that support piggybacking the initial mechToken,
     an optimistic negotiation response is possible and includes in that
     case a responseToken which may continue the authentication exchange
     (e.g. when mutual authentication has been requested or when
     unilateral authentication requires several round trips). Otherwise

the responseToken is used to carry the tokens specific to the
mechanism selected. For subsequent tokens (if any) returned by the
target, negResult, and supportedMech are not present.

For the last token returned by the target, the mechListMIC, when
present, is a MIC computed over the MechTypes using the selected
mechanism.

negResult
     Result of the negotiation exchange, specified by the target.

     This can be either :

          accept_completed
               The target accepts the preferred security mechanism,
                and the context is established for the target or,

          accept_incomplete
               The target accepts one of the proposed security
               mechanisms and further exchanges are necessary, or,

          reject
               The target rejects all the proposed security
               mechanisms.

supportedMech
     This field has to be present when negResult is "accept_completed"
     or "accept_incomplete". It is a choice from the mechanisms offered
     by the initiator.

responseToken
     This field may be used either to transmit the response to the
     mechToken when sent by the initiator and when the first mechanism
     from the list has been selected by the target or to carry the
     tokens specific to the selected security mechanism.

mechListMIC
     If the selected mechanism is capable of integrity protection, this
     field must be present in the last message of the negotiation,
     (i.e., when the underlying mechanism returns a non-empty token and
     a major status of GSS_S_COMPLETE); it contains the result of a
     GetMIC of the MechTypes field in the initial NegTokenInit. It
     allows to verify that the list initially sent by the initiator has
     been received unmodified by the target.

3.2.2. Processing of mechListMIC.

   If the mechanism selected by the negotiation does not support
   integrity, then no mechListMIC is included, otherwise a mechListMIC
   must be used and validated as indicated below.

   If the mechanism supports integrity and uses an even number of
   messages, then the target must compute a MIC as described above, and
   send this in the final NegTokenTarg along with the final mechToken.
   The initiator when receiving the last token must require that the
   mechListMIC field be present and valid. In the absence of a valid
   mechListMIC, the negotiation must fail as if the last context
   establishment token was invalid.

   In the case that the chosen mechanism supports integrity and uses an
   odd number of messages, the final mechanism token will be sent from
   the initiator to the target. In this case, there is a tradeoff
   between using the optimal number of messages, or using an additional
   message from the target to the initiator in order to give the
   initiator assurance that no modification of the initiator's mechanism
   list occurred. The implementation can choose which tradeoff to make.

   When generating the final NegTokenInit message, the NegTokenInit may
   optionally include a mechListMIC which is the result of a GetMIC of
   the MechTypes in the initial NegTokenInit and return GSS_S_COMPLETE.
   The target must check the presence of the MIC computed over the
   mechList sent in the initial NegTokenInit. Three cases may then be
   considered:

      1) If the mechListMIC is present and correct, then
         GSS_S_COMPLETE is returned to the target with no token; the
         context is established by the target.

      2) If the mechListMIC is present but invalid, then the context
         establishment must fail. An error major status code is
         returned to the target.

      3) If the mechListMIC is not included in the final NegTokenInit,
         then GSS_S_COMPLETE must be returned to the target with a
         token. This token must be a NegTokenTarg, with a MIC included
         as described above, and no responseToken. The application will
         then send this token back to the initiator, which must verify
         that the mechListMIC field is present and valid.

          Note: If the MIC was originally sent by the initiator, but
                thenafter deleted by an attacker, the target will send
                back a token according to the description above, but the
                initiator will be unable to process that returned token
                and the context establishment must then fail.

4.   EXAMPLES : SECURITY MECHANISM NEGOTIATION

   Here are some examples of security mechanism negotiation between an
   initiator (I) and a target (T).

4.1.   Initial steps

   (I) supports two security mechanism types (GSS-MECH1 and GSS-MECH2).

   (I) invokes GSS_Init_sec_context() with :

   Input
     mech_type = OID for negotiation mechanism or NULL, if the
     negotiation mechanism is the default mechanism.

   Output
     major_status = GSS_S_CONTINUE_NEEDED
     output_token = negTokenInit

   The negotiation token (negTokenInit) contains two security mechanisms
   with :
     mechType = GSS-MECH1 or
     mechType = GSS-MECH2

   (I) sends to (T) the negotiation token.

4.2 Successful negotiation steps

   (T) supports GSS-MECH2
   (T) receives the negotiation token (negTokenInit) from (I)
   (T) invokes GSS_Accept_sec_context() with :

   Input
        input_token = negTokenInit

   Output
        major_status = GSS_S_CONTINUE_NEEDED
        output_token = negTokenTarg

   The negotiation token (negTokenTarg) contains :
        negResult = accept (the negotiation result)
        supportedMech : mechType = GSS-MECH2

```
    (T) returns the negotiation token (negTokenTarg) to (I)
    (I) invokes GSS_Init_sec_context() with :

    Input
        input_token = negTokenTarg

    Output
        major_status = GSS_S_COMPLETE
        output_token = initialContextToken (initial context token
                                            for GSS-MECH2)
        mech_type = GSS-MECH2
```

The subsequent steps are security mechanism specific, and work as
specified in [1]. The output tokens from the security mechanism are
encapsulated in a NegTokenTarg message (with the supportedMech field
omitted, and the mechListMIC included with the last token).

4.3. Failed negotiation steps

```
    (T) supports GSS-MECH3.
    (T) receives the negotiation token (negTokenInit) from (I)
    (T) invokes GSS_Accept_sec_context() with :

    Input
        input_token = negTokenInit

    Output
        major_status = GSS_S_BAD_MECH
        output_token = negTokenTarg
```

The negotiation token (negTokenTarg) contains :

```
        negResult = reject (the negotiation result)
```

```
    (T) returns the negotiation token (negTokenTarg) to (I)
    (I) invokes GSS_Init_sec_context() with :

    Input
        input_token = negTokenTarg

    Output
        major_status = GSS_S_BAD_MECH
```

The security context establishment has failed.

4.4 Successful Negotiation with preferred mechanism info

   (I) supports two security mechanism types (GSS-MECH1 and GSS-MECH2).

   (I) invokes GSS_Init_sec_context() with :

   Input
        mech_type = OID for negotiation mechanism or NULL, if the
        negotiation mechanism is the default mechanism.

   Output
        major_status = GSS_S_CONTINUE_NEEDED
        output_token = negTokenInit

   The negotiation token (negTokenInit) contains two security mechanisms
   with :
        mechType = GSS-MECH1 or
        mechType = GSS-MECH2

        mechToken = output_token from GSS_Init_sec_context
       (first mechType) as described in [1]

   (I) sends to (T) the negotiation token.

   (T) supports GSS-MECH1.
   (T) receives the negotiation token (negTokenInit) from (I)
   (T) invokes GSS_Accept_sec_context() with :

   Input
        input_token = negTokenInit

   Output
        major_status = GSS_S_CONTINUE_NEEDED
        output_token = negTokenTarg

   The negotiation token (negTokenTarg) contains :
        negResult = accept (the negotiation result)
        supportedMech : mechType = GSS-MECH1

        mechToken = output_token from
                        GSS_Accept_sec_context(mechToken)

   (T) returns the negotiation token (negTokenTarg) to (I)
   (I) invokes GSS_Init_sec_context() with :

   Input
        input_token = negTokenTarg

```
Output
     major_status = GSS_S_COMPLETE or GSS_S_CONTINUE_NEEDED as needed
     output_token = ContextToken (initial or subsequent context token
                    for GSS-MECH1)
     mech_type = GSS-MECH1
```

Specific implementations of the protocol can support the optimistic
negotiation by completing the security context establishment using the
agreed upon mechanism as described in [1]. As described above in
section 5.2, the output tokens from the security mechanisms are
encapsulated in a NegTokenTarg message (with the negResult and
supportedMech fields omitted, and the mechListMIC included with the
last token).

5. SECURITY CONSIDERATIONS

   When the mechanism selected by the target from the list supplied by
   the initiator supports integrity protection, then the negotiation is
   protected.

   When one of the mechanisms proposed by the initiator does not support
   integrity protection, then the negotiation is exposed to all threats
   a non secured service is exposed. In particular, an active attacker
   can force to use a security mechanism which is not the common
   preferred one (when multiple security mechanisms are shared between
   peers) but which is acceptable anyway to the target.

   In any case, the communicating peers may be exposed to the denial of
   service threat.

6. ACKNOWLEDGMENTS

   Acknowledgments are due to Stephen Farrell of SSE, Marc Horowitz of
   Stonecast, John Linn of RSA Laboratories, Piers McMahon of Platinum
   Technology, Tom Parker of ICL and Doug Rosenthal of EINet, for
   reviewing earlier versions of this document and for providing useful
   inputs. Acknowledgments are also due to Peter Brundrett of Microsoft
   for his proposal for an optimistic negotiation, and for Bill
   Sommerfeld of Epilogue Technology for his proposal for protecting the
   negotiation.

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:

   GSS_Get_neg_mechs() indicates the set of security mechanisms
   available on the local system to the caller for negotiation.

   GSS_Set_neg_mechs() specifies the set of security mechanisms to be
   used on the local system by the caller for negotiation.

A.1. GSS_Set_neg_mechs call

   Input:
        cred_handle   CREDENTIAL HANDLE
                             - NULL specifies default credentials
        mech_set   SET OF OBJECT IDENTIFIER

   Outputs:
        major_status INTEGER,
        minor_status INTEGER,

   Return major_status codes :
     GSS_S_COMPLETE indicates that the set of security mechanisms
     available for negotiation has been set to mech_set. 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 specialised 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 mechanism preference for the target.

A.2. GSS_Get_neg_mechs call

     Input:
          cred_handle   CREDENTIAL HANDLE
                          - NULL specifies default credentials

     Outputs:
          major_status INTEGER,
          minor_status INTEGER,
          mech_set   SET OF OBJECT IDENTIFIER

     Return major_status codes :
          GSS_S_COMPLETE indicates that the set of security mechanisms
          available for negotiation has been returned in
          mech_option_set.
          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 specialised 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.

REFERENCES

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

     [2] Standard ECMA-206, "Association Context Management including
         Security Context Management", December 1993. Available on
         http://www.ecma.ch

AUTHORS' ADDRESSES

   Eric Baize
   Bull - 300 Concord Road
   Billerica, MA 01821 - USA

   Phone: +1 978 294 61 37
   Fax: +1 978 294 61 09
   EMail: Eric.Baize@bull.com

   Denis Pinkas
   Bull
   Rue Jean-Jaures
   BP 68
   78340 Les Clayes-sous-Bois - FRANCE

   Phone: +33 1 30 80 34 87
   Fax: +33 1 30 80 33 21
   EMail: Denis.Pinkas@bull.net

Full Copyright Statement