Initial and Pass Through Authentication Using Kerberos V5 and the GSS-API (IAKERB)
draft-ietf-kitten-iakerb-03

Abstract

   This document defines extensions to the Kerberos protocol and the
   GSS-API Kerberos mechanism that enable a GSS-API Kerberos client to
   exchange messages with the KDC by using the GSS-API acceptor as a
   proxy, encapsulating the Kerberos messages inside GSS-API tokens.
   With these extensions a client can obtain Kerberos tickets for
   services where the KDC is not accessible to the client, but is
   accessible to the application server.

Table of Contents

1. Introduction

When authenticating using Kerberos V5, clients obtain tickets from a
KDC and present them to services. This model of operation cannot
work if the client does not have access to the KDC. For example, in
remote access scenarios, the client must initially authenticate to an
access point in order to gain full access to the network. Here the
client may be unable to directly contact the KDC either because it
does not have an IP address, or the access point packet filter does
not allow the client to send packets to the Internet before it
authenticates to the access point. The Initial and Pass Through
Authentication Using Kerberos (IAKERB) mechanism allows for the use
of Kerberos in such scenarios where the client is unable to directly
contact the KDC, by using the service to pass messages between the
client and the KDC. This allows the client to obtain tickets from
the KDC and present them to the service, as in normal Kerberos
operation.

Recent advancements in extending Kerberos permit Kerberos
authentication to complete with the assistance of a proxy. The

Kerberos [RFC4120] pre-authentication framework [RFC6113] prevents
the exposure of weak client keys over the open network. The Kerberos
support of anonymity [RFC6112] provides for privacy and further
complicates traffic analysis. The kdc-referrals option defined in
[RFC6113] may reduce the number of messages exchanged while obtaining
a ticket to exactly two even in cross-realm authentications.

Building upon these Kerberos extensions, this document extends
[RFC4120] and [RFC4121] such that the client can communicate with the
KDC using a Generic Security Service Application Program Interface
(GSS-API) [RFC2743] acceptor as a message-passing proxy. (This is
completely unrelated to the type of proxy specified in [RFC4120].)
The client acts as a GSS-API initiator, and the GSS-API acceptor
relays the KDC request and reply messages between the client and the
KDC, transitioning to normal [RFC4121] GSS-krb5 messages once the
client has obtained the necessary credentials. Consequently, IAKERB
as defined in this document requires the use of the GSS-API.

The GSS-API acceptor, when relaying these Kerberos messages, is
called an IAKERB proxy.

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. GSS-API Encapsulation

The GSS-API mechanism Objection Identifier (OID) for IAKERB is id-
kerberos-iakerb:

```
id-kerberos-iakerb ::=
{ iso(1) org(3) dod(6) internet(1) security(5) kerberosV5(2)
iakerb(5) }
```

All context establishment tokens of IAKERB MUST have the token
framing described in section 4.1 of [RFC4121] with the mechanism OID
being id-kerberos-iakerb. MIT implemented an earlier draft of this
specification; details on how to interoperate with that
implementation can be found in Appendix A.

The client starts by constructing a ticket request, as if it is being
made directly to the KDC. Instead of contacting the KDC directly,
the client encapsulates the request message into the output token of
the GSS_Init_security_context() call and returns
GSS_S_CONTINUE_NEEDED [RFC2743], indicating that at least one more
token is required in order to establish the context. The output

token is then passed over the application protocol for use as the
input token to the GSS_Accept_sec_context() call in accordance with
GSS-API. The GSS-API acceptor extracts the Kerberos request from the
input token, locates the target KDC, and sends the request on behalf
of the client. After receiving the KDC reply, the GSS-API acceptor
then encapsulates the reply message into the output token of
GSS_Accept_sec_context(). The GSS-API acceptor returns
GSS_S_CONTINUE_NEEDED [RFC2743] indicating that at least one more
token is required in order to establish the context. The output
token is passed to the initiator over the application protocol in
accordance with GSS-API.

```
      +--------+              +--------------+           +-----+
      |        | -------->  |              | -------->  |     |
      | Client |            | IAKERB proxy |            | KDC |
      |        | <-------- |              | <-------- |     |
      +--------+              +--------------+           +-----+
```

For all context tokens generated by the IAKERB mechanism, the
innerToken described in section 4.1 of [RFC4121] has the following
format: it starts with a two-octet token-identifier (TOK_ID), which
is followed by an IAKERB message or a Kerberos message.

Only one IAKERB specific message, namely the IAKERB_PROXY message, is
defined in this document. The TOK_ID values for Kerberos messages
are the same as defined in [RFC4121].

```
          Token            TOK_ID Value in Hex
          --------------------------------------
          IAKERB_PROXY          05 01
```

The content of the IAKERB_PROXY message is defined as an IAKERB-
HEADER structure immediately followed by a Kerberos message, which is
optional. The Kerberos message can be an AS-REQ, an AS-REP, a TGS-
REQ, a TGS-REP, or a KRB-ERROR as defined in [RFC4120].

```
      IAKERB-HEADER ::= SEQUENCE {
          -- Note that the tag numbers start at 1, not 0, which would
          -- be more conventional for Kerberos.
          target-realm      [1] UTF8String,
              -- The name of the target realm.
          cookie            [2] OCTET STRING OPTIONAL,
              -- Opaque data, if sent by the server,
              -- MUST be copied by the client verbatim into
              -- the next IAKRB_PROXY message.
          ...
      }
```

The IAKERB-HEADER structure and all the Kerberos messages MUST be
encoded using Abstract Syntax Notation One (ASN.1) Distinguished
Encoding Rules (DER) [CCITT.X680.2002] [CCITT.X690.2002].

The client fills out the IAKERB-HEADER structure as follows: the
target-realm contains the realm name the ticket request is addressed
to. In the initial message from the client, the cookie field is
absent. The client MAY send a completely empty IAKERB_PROXY message
(consisting solely of the octets 05 01 and an IAKERB_HEADER with
zero-length target-realm) in order to query the Kerberos realm of the
acceptor, see Section 3.1. In all other cases, the client MUST
specify a target-realm. This can be the realm of the client's host,
if no other realm information is available. client's host.

Upon receipt of the IAKERB_PROXY message, the GSS-API acceptor
inspects the target-realm field in the IAKERB_HEADER, locates a KDC
for that realm, and sends the ticket request to that KDC. The IAKERB
proxy MAY engage in fallback behavior, retransmitting packets to a
given KDC and/or sending the request to other KDCs in that realm if
the initial transmission does not receive a reply, as would be done
if the proxy was making requests on its own behalf.

The GSS-API acceptor encapsulates the KDC reply message in the
returned IAKERB message. It fills out the target realm using the
realm sent by the client and the KDC reply message is included
immediately following the IAKERB-HEADER header.

When the GSS-API acceptor is unable to obtain an IP address for a KDC
in the client's realm, it sends a KRB_ERROR message with the code
KRB_AP_ERR_IAKERB_KDC_NOT_FOUND to the client in place of an actual
reply from the KDC, and the context fails to establish. There is no
accompanying error data defined in this document for this error code.

        KRB_AP_ERR_IAKERB_KDC_NOT_FOUND   85
             -- The IAKERB proxy could not find a KDC.

When the GSS-API acceptor has an IP address for at least one KDC in
the target realm, but does not receive a response from any KDC in the
realm (including in response to retries), it sends a KRB_ERROR
message with the code KRB_AP_ERR_IAKERB_KDC_NO_RESPONSE to the client
and the context fails to establish. There is no accompanying error
data defined in this document for this error code.

        KRB_AP_ERR_IAKERB_KDC_NO_RESPONSE   86
             -- The KDC did not respond to the IAKERB proxy.

The IAKERB proxy can send opaque data in the cookie field of the
IAKERB-HEADER structure in the server reply to the client, in order

to, for example, minimize the amount of state information kept by the
GSS-API acceptor. The content and the encoding of the cookie field
is a local matter of the IAKERB proxy. Whenever the cookie is
present in a token received by the initiator, the initiator MUST copy
the cookie verbatim into its subsequent response tokens which contain
IAKERB_PROXY messages.

The client and the server can repeat the sequence of sending and
receiving the IAKERB messages as described above for an arbitrary
number of message exchanges, in order to allow the client to interact
with the KDC through the IAKERB proxy, and to obtain Kerberos tickets
as needed to authenticate to the acceptor.

Once the client has obtained the service ticket needed to
authenticate to the acceptor, subsequent GSS-API context tokens are
of type KRB_AP_REQ, not IAKERB_PROXY, and the client performs the
client-server application exchange as defined in [RFC4120] and
[RFC4121].

For implementations conforming to this specification, both the
authenticator subkey and the GSS_EXTS_FINISHED extension as defined
in Section 4 MUST be present in the AP-REQ authenticator. This
checksum provides integrity protection for the IAKERB messages
previously exchanged, including the unauthenticated clear texts in
the IAKERB-HEADER structure.

If the pre-authentication data is encrypted in the long-term
password-based key of the principal, the risk of security exposures
is significant. Implementations SHOULD utilize the AS_REQ armoring
as defined in [RFC6113] unless an alternative protection is deployed.
In addition, the anonymous Kerberos FAST option is RECOMMENDED for
the client to complicate traffic analysis.

3.1. Enterprise principal names

The introduction of principal name canonicalization by [RFC6806]
created the possibility for a client to have a principal name (of
type NT-ENTERPRISE) for which it is trying to obtain credentials, but
no information about what realm's KDC to contact to obtain those
credentials. A Kerberos client not using IAKERB would typically
resolve the NT-ENTERPRISE name to a principal name by starting from
the realm of the client's host and finding out the true realm of the
enterprise principal based on referrals [RFC6806].

A client using IAKERB may not have any realm information, even for
the realm of the client's host, or may know that the client host's
realm is not appropriate for a given enterprise principal name. In
such cases, the client can retrieve the realm of the GSS-API acceptor

as follows: the client returns GSS_S_CONTINUE_NEEDED with the output
token containing an IAKERB message with an empty target-realm in the
IAKERB-HEADER and no Kerberos message following the IAKERB-HEADER
structure. Upon receipt of the realm request, the GSS-API acceptor
fills out an IAKERB_PROXY response message, filling the target-realm
field with the realm of the acceptor, and returns
GSS_S_CONTINUE_NEEDED with the output token containing the IAKERB
message with the server's realm and no Kerberos message following the
IAKERB-HEADER header. The GSS-API initiator can then use the
returned realm in subsequent IAKERB messages to resolve the NT-
ENTERPRISE name type. Since the GSS-API acceptor can act as a
Kerberos acceptor, it always has an associated Kerberos realm.

4. Finish Message

For implementations conforming to this specification, the
authenticator subkey in the AP-REQ MUST alway be present, and the
Exts field in the GSS-API authenticator [RFC6542] MUST contain an
extension of type GSS_EXTS_FINISHED with extension data containing
the ASN.1 DER encoding of the structure KRB-FINISHED.

```
   GSS_EXTS_FINISHED             2
        --- Data type for the IAKERB checksum.

   KRB-FINISHED ::= {
        -- Note that the tag numbers start at 1, not 0, which would be
        -- more conventional for Kerberos.
        gss-mic [1] Checksum,
            -- Contains the checksum [RFC3961] of the GSS-API tokens
            -- exchanged between the initiator and the acceptor,
            -- and prior to the containing AP_REQ GSS-API token.
            -- The checksum is performed over the GSS-API tokens
            -- exactly as they were transmitted and received,
            -- in the order that the tokens were sent.
         ...
   }
```

The gss-mic field in the KRB-FINISHED structure contains a Kerberos
checksum [RFC3961] of all the preceding context tokens of this GSS-
API context (including the generic token framing of the GSSAPI-Token
type from [RFC4121]), concatenated in chronological order (note that
GSS-API context token exchanges are synchronous). The checksum type
is the required checksum type of the enctype of the subkey in the
authenticator, the protocol key for the checksum operation is the
authenticator subkey, and the key usage number is KEY_USAGE_FINISHED.

```
           KEY_USAGE_FINISHED            41
```

The GSS-API acceptor MUST then verify the checksum contained in the
GSS_EXTS_FINISHED extension. This checksum provides integrity
protection for the messages exchanged including the unauthenticated
clear texts in the IAKERB-HEADER structure.

5. Addresses in Tickets

In IAKERB, the machine sending requests to the KDC is the GSS-API
acceptor and not the client. As a result, the client should not
include its addresses in any KDC requests for two reasons. First,
the KDC may reject the forwarded request as being from the wrong
client. Second, in the case of initial authentication for a dial-up
client, the client machine may not yet possess a network address.
Hence, as allowed by [RFC4120], the addresses field of the AS-REQ and
TGS-REQ requests SHOULD be blank.

6. Security Considerations

The IAKERB proxy is a man-in-the-middle for the client's Kerberos
exchanges. The Kerberos protocol is designed to be used over an
untrusted network, so this is not a critical flaw, but it does expose
to the IAKERB proxy all information sent in cleartext over those
exchanges, such as the principal names in requests. Since the
typical usage involves the client obtaining a service ticket for the
service operating the proxy, which will receive the client principal
as part of normal authentication, this is also not a serious concern.
However, an IAKERB client not using an armored FAST channel [RFC6113]
sends an AS_REQ with pre-authentication data encrypted in the long-
term keys of the user, even before the acceptor is authenticated.
This subjects the user's long-term key to an offline attack by the
proxy. To mitigate this threat, the client SHOULD use FAST [RFC6113]
and its KDC authentication facility to protect the user's
credentials.

Similarly, the client principal name is in cleartext in the AS and
TGS exchanges, whereas in the AP exchanges embedded in GSS context
tokens for the regular krb5 mechanism, the client principal name is
present only in encrypted form. Thus, more information is exposed
over the network path between initiator and acceptor when IAKERB is
used than when the krb5 mechanism is used, unless FAST armor is
employed. (This information would be exposed in other traffic from
the initiator when the krb5 mech is used.) As such, to complicate
traffic analysis and provide privacy for the client, the client
SHOULD request the anonymous Kerberos FAST option [RFC6113].

Similar to other network access protocols, IAKERB allows an
unauthenticated client (possibly outside the security perimeter of an
organization) to send messages that are proxied to servers inside the

perimeter. To reduce the attack surface, firewall filters can be
applied to restrict from which hosts client requests can be proxied,
and the proxy can further restrict the set of realms to which
requests can be proxied.

In the intended use scenario, the client uses the proxy to obtain a
TGT and then a service ticket for the service it is authenticating to
(possibly preceeded by exchanges to produce FAST armor). However,
the protocol allows arbitrary KDC-REQs to be passed through, and
there is no limit to the number of exchanges that may be proxied.
The client can send KDC-REQs unrelated to the current authentication,
and obtain service tickets for other service principals in the
database of the KDC being contacted.

In a scenario where DNS SRV RR's are being used to locate the KDC,
IAKERB is being used, and an external attacker can modify DNS
responses to the IAKERB proxy, there are several countermeasures to
prevent arbitrary messages from being sent to internal servers:

1. KDC port numbers can be statically configured on the IAKERB
    proxy. In this case, the messages will always be sent to KDC's.
    For an organization that runs KDC's on a static port (usually
    port 88) and does not run any other servers on the same port,
    this countermeasure would be easy to administer and should be
    effective.

2. The proxy can do application level sanity checking and filtering.
    This countermeasure should eliminate many of the above attacks.

3. DNS security can be deployed. This countermeasure is probably
    overkill for this particular problem, but if an organization has
    already deployed DNS security for other reasons, then it might
    make sense to leverage it here. Note that Kerberos could be used
    to protect the DNS exchanges. The initial DNS SRV KDC lookup by
    the proxy will be unprotected, but an attack here is at most a
    denial of service (the initial lookup will be for the proxy's KDC
    to facilitate Kerberos protection of subsequent DNS exchanges
    between itself and the DNS server).

7. Acknowledgements

Jonathan Trostle, Michael Swift, Bernard Aboba and Glen Zorn wrote
earlier revision of this document.

The hallway conversations between Larry Zhu and Nicolas Williams
formed the basis of this document.

8. Assigned Numbers

   The value for the error code KRB_AP_ERR_IAKERB_KDC_NOT_FOUND is 85.

   The value for the error code KRB_AP_ERR_IAKERB_KDC_NO_RESPONSE is 86.

   The key usage number KEY_USAGE_FINISHED is 41.

   The key usage number KEY_USAGE_IAKERB_FINISHED is 42.

9. IANA Considerations

   IANA is requested to make a modification in the "Kerberos GSS-API
   Token Type Identifiers" registry.

   The following data to the table:

```
               +-------+--------------+------------+
               | ID    | Description  | Reference  |
               +-------+--------------+------------+
               | 05 01 | IAKERB_PROXY | [THIS RFC] |
               +-------+--------------+------------+
```

10. References

10.1. Normative References

   [CCITT.X680.2002]
            International Telephone and Telegraph Consultative
            Committee, "Abstract Syntax Notation One (ASN.1):
            Specification of basic notation", CCITT Recommendation
            X.680, July 2002.

   [CCITT.X690.2002]
            International Telephone and Telegraph Consultative
            Committee, "ASN.1 encoding rules: Specification of basic
            encoding Rules (BER), Canonical encoding rules (CER) and
            Distinguished encoding rules (DER)", CCITT Recommendation
            X.690, July 2002.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119,
            DOI 10.17487/RFC2119, March 1997,
            <http://www.rfc-editor.org/info/rfc2119>.

[RFC2743] Linn, J., "Generic Security Service Application Program
           Interface Version 2, Update 1", RFC 2743,
           DOI 10.17487/RFC2743, January 2000,
           <http://www.rfc-editor.org/info/rfc2743>.

[RFC3961] Raeburn, K., "Encryption and Checksum Specifications for
           Kerberos 5", RFC 3961, DOI 10.17487/RFC3961, February
           2005, <http://www.rfc-editor.org/info/rfc3961>.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The
           Kerberos Network Authentication Service (V5)", RFC 4120,
           DOI 10.17487/RFC4120, July 2005,
           <http://www.rfc-editor.org/info/rfc4120>.

[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,
           DOI 10.17487/RFC4121, July 2005,
           <http://www.rfc-editor.org/info/rfc4121>.

[RFC6542] Emery, S., "Kerberos Version 5 Generic Security Service
           Application Program Interface (GSS-API) Channel Binding
           Hash Agility", RFC 6542, DOI 10.17487/RFC6542, March 2012,
           <http://www.rfc-editor.org/info/rfc6542>.

10.2. Informative references

[RFC6112] Zhu, L., Leach, P., and S. Hartman, "Anonymity Support for
           Kerberos", RFC 6112, DOI 10.17487/RFC6112, April 2011,
           <http://www.rfc-editor.org/info/rfc6112>.

[RFC6113] Hartman, S. and L. Zhu, "A Generalized Framework for
           Kerberos Pre-Authentication", RFC 6113,
           DOI 10.17487/RFC6113, April 2011,
           <http://www.rfc-editor.org/info/rfc6113>.

[RFC6806] Hartman, S., Ed., Raeburn, K., and L. Zhu, "Kerberos
           Principal Name Canonicalization and Cross-Realm
           Referrals", RFC 6806, DOI 10.17487/RFC6806, November 2012,
           <http://www.rfc-editor.org/info/rfc6806>.

Appendix A. Interoperate with Previous MIT version

MIT implemented an early draft version of this document. This
section gives a method for detecting and interoperating with that
version.

Initiators behave as follows:

o If the first acceptor token begins with generic token framing as
   described in [section 3.1 of [RFC2743]](), then use the protocol as
   defined in this document.

o If the first acceptor token is missing the generic token framing
   (i.e., the token begins with the two-byte token ID 05 01), then

   * When creating the finish message, the value of one (1) should
      be used in place of GSS_EXTS_FINISHED.

   * When computing the checksum, the value of
      KEY_USAGE_IAKERB_FINISHED should be used in place of
      KEY_USAGE_FINISHED.

         KEY_USAGE_IAKERB_FINISHED   42

Acceptors behave as follows:

o After the first initiator token, allow initiator tokens to omit
   generic token framing. This allowance is required only for
   IAKERB_PROXY messages (those using token ID 05 01), not for tokens
   defined in [[RFC4121]()].

o If the AP-REQ authenticator contains an extension of type 1
   containing a KRB-FINISHED message, then process the extension as
   if it were of type GSS_EXTS_FINISHED, except with a key usage of
   KEY_USAGE_IAKERB_FINISHED (42) instead of KEY_USAGE_FINISHED (41).

Authors' Addresses

Benjamin Kaduk (editor)
Akamai

Email: kaduk@mit.edu

Jim Schaad (editor)
Soaring Hawk Consulting

Email: ietf@augustcellars.com

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

Email: lzhu@microsoft.com

Jeffery Altman
Secure Endpoints
255 W 94th St
New York, NY 10025
US

Email: jaltman@secure-endpoints.com