INTERNET-DRAFT   Mike Swift
draft-ietf-cat-iakerb-07.txt   University of WA
Updates: RFC 1510, 1964   Jonathan Trostle
July 2001   Cisco Systems
                                                  Bernard Aboba
                                                  Microsoft
                                                  Glen Zorn
                                                  Cisco Systems

Extending the GSS Kerberos Mechanism for Initial Kerberos Authentication (IAKERB)

<draft-ietf-cat-iakerb-07.txt>

1. Abstract

This document defines extensions to the Kerberos protocol specification (RFC 1510 [1]) and GSSAPI Kerberos mechanism (RFC 1964 [2]) that enables a RFC 1964 client to obtain Kerberos tickets for services where the KDC is not accessible to the client, but is accessible to the application server. Some common scenarios where lack of accessibility would occur are when the client does not have an IP address prior to authenticating to an access point, the client is unable to locate a KDC, or a KDC is behind a firewall. The document specifies two protocols to allow a client to exchange KDC

messages (which are GSS encapsulated) with an IAKERB proxy instead of a KDC.

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 [7].

3. Motivation

When authenticating using Kerberos V5, clients obtain tickets from a KDC and present them to services. This method of operation works well in many situations, but is not always applicable. The following is a list of some of the scenarios that this proposal addresses:

(1) 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.

(2) A KDC is behind a firewall so the client will send Kerberos messages to the IAKERB proxy which will transmit the KDC request and reply messages between the client and the KDC. (The IAKERB proxy is a special type of Kerberos application server that also relays KDC request and reply messages between a client and the KDC).

4. Overview

This proposal specifies two protocols that address the above scenarios: the IAKERB proxy option and the IAKERB minimal messages option. In the IAKERB proxy option (see Figure 1) an application server called the IAKERB proxy acts as a protocol gateway and proxies Kerberos messages back and forth between the client and the KDC. The IAKERB proxy is also responsible for locating the KDC and may additionally perform other application proxy level functions such as auditing.

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

Figure 1: IAKERB proxying

The second protocol is the minimal messages protocol that extends the technique in [5]; this protocol is targetted at environments where the number of messages (prior to key establishment) needs to be minimized. Here the client sends its ticket granting ticket (TGT) to the IAKERB proxy (in a KRB_TKT_PUSH message) for the TGS case. The

IAKERB proxy then sends a TGS_REQ to the KDC with the client's TGT in the additional tickets field of the TGS_REQ message. As a result, the returned ticket will list the client as the ticket's server principal, and will be encrypted with the session key from the client's TGT. The IAKERB proxy then uses this ticket to generate an

AP request that is sent to the client (see Figure 2). Thus mutual authentication is accomplished with three messages between the client and the IAKERB proxy versus four or more (the difference is larger if crossrealm operations are involved). Subsequent to mutual authentication and key establishment, the IAKERB proxy sends a ticket to the client (in a KRB_TKT_PUSH message) that contains the same fields as the original service ticket except the client and server names are reversed and it is encrypted in a long term key known to the IAKERB proxy. Its purpose is to enable fast subsequent re-authentication by the client to the application server (using the conventional AP request AP reply exchange) for subsequent sessions. In addition to minimizing the number of messages, a secondary goal is to minimize the number of bytes transferred between the client and the IAKERB proxy prior to mutual authentication and key establishment. Therefore, the final service ticket (the reverse ticket) is sent after mutual authentication and key establishment is complete, rather than as part of the initial AP_REQ from the IAKERB proxy to the client.

The AS_REQ case for the minimal messages option is similar, where the client sends up the AS_REQ message and the IAKERB proxy forwards it to the KDC. The IAKERB proxy pulls the client TGT out of the AS_REP message and also forwards the AS_REP message back to the client. The protocol now proceeds as in the TGS_REQ case with the IAKERB proxy including the client's TGT in the additional tickets field of the TGS_REQ message.

```
     Client  --------> IAKERB proxy
             TKT_PUSH (w/ TGT)

     Client            IAKERB proxy --------------------> KDC
                                     TGS_REQ with client
                                     TGT as additional TGT

     Client            IAKERB proxy <-------------------- KDC
                                     TGS_REP with service
                                     ticket

     Client <--------  IAKERB proxy                       KDC
              AP_REQ

     Client -------->  IAKERB proxy                       KDC
              AP_REP

-------------------------------------------------------------
 post-key establishment and application data flow phase:

     Client <--------  IAKERB proxy                       KDC
```

```
                        TKT_PUSH (w/ticket targetted at IAKERB proxy
                               to enable fast subsequent authentication)
```

Figure 2: IAKERB Minimal Messages Option: TGS case

A compliant IAKERB proxy MUST implement the IAKERB proxy protocol, and MAY implement the IAKERB minimal message protocol. In general, the existing Kerberos paradigm where clients contact the KDC to obtain service tickets should be preserved where possible.

If the client has a service ticket for the target server, needs to authenticate to the target server, and does not have direct connectivity with the target server, it should use the IAKERB proxy protocol. If the client needs to obtain a crossrealm TGT (and the conventional Kerberos protocol cannot be used), then the IAKERB proxy protocol must be used. In a scenario where the client does not have a service ticket for the target server, it is crucial that the number of messages between the client and the target server be minimized (especially if the client and target server are in different realms), and/or it is crucial that the number of bytes transferred between the client and the target server be minimized, then the client should consider using the minimal messages protocol. The reader should see the security considerations section regarding the minimal messages protocol.

5. GSSAPI Encapsulation

The mechanism ID for IAKERB proxy GSS-API Kerberos, in accordance with the mechanism proposed by SPNEGO [8] for negotiating protocol variations, is: {iso(1) org(3) dod(6) internet(1) security(5) mechanisms(5) iakerb(10) iakerbProxyProtocol(1)}. The proposed mechanism ID for IAKERB minimum messages GSS-API Kerberos, in accordance with the mechanism proposed by SPNEGO for negotiating protocol variations, is: {iso(1) org(3) dod(6) internet(1) security(5) mechanisms(5) iakerb(10) iakerbMinimumMessagesProtocol(2)}.

The AS request, AS reply, TGS request, and TGS reply messages are all encapsulated using the format defined by RFC1964 [2]. This consists of the GSS-API token framing defined in appendix B of RFC1508 [3]:

```
InitialContextToken ::= [APPLICATION 0] IMPLICIT SEQUENCE {
        thisMech        MechType
                -- MechType is OBJECT IDENTIFIER
                -- representing "Kerberos V5"
        innerContextToken ANY DEFINED BY thisMech
                -- contents mechanism-specific;
                -- ASN.1 usage within innerContextToken
                -- is not required
     }
```

The innerContextToken consists of a 2-byte TOK_ID field (defined below), followed by the Kerberos V5 KRB_AS_REQ, KRB_AS_REP, KRB_TGS_REQ, or KRB_TGS_REP messages, as appropriate. The TOK_ID field shall be one of the following values, to denote that the message is either a request to the KDC or a response from the KDC.

Message	TOK_ID
KRB_KDC_REQ	00 03
KRB_KDC_REP	01 03

We also define the token ID for the KRB_TKT_PUSH message (defined below and used in the minimal messages variation):

Message	TOK_ID
KRB_TKT_PUSH	02 03

For completeness, we list the other RFC 1964 defined token ID's here:

Message	TOK_ID
AP_REQ	01 00
AP_REP	02 00
KRB_ERROR	03 00

6. The IAKERB proxy protocol

The IAKERB proxy will proxy Kerberos KDC request, KDC reply, and KRB_ERROR messages back and forth between the client and the KDC as illustrated in Figure 1. Messages received from the client must first have the Kerberos GSS header (RFC1964 [2]) stripped off. The unencapsulated message will then be forwarded to a KDC. The IAKERB proxy is responsible for locating an appropriate KDC using the realm information in the KDC request message it received from the client. In addition, the IAKERB proxy SHOULD implement a retry algorithm for KDC requests over UDP (including selection of alternate KDC's if the initial KDC does not respond to its requests). For messages sent by the KDC, the IAKERB proxy encapsulates them with a Kerberos GSS header before sending them to the client.

We define two new Kerberos error codes that allow the proxy to indicate the following error conditions to the client:

(a) when the proxy is unable to obtain an IP address for a KDC in the client's realm, it sends the KRB_IAKERB_ERR_KDC_NOT_FOUND KRB_ERROR (80) message to the client.

(b) when the proxy has an IP address for a KDC in the client realm, but does not receive a response from any KDC in the realm (including in response to retries), it sends the KRB_IAKERB_ERR_KDC_NO_RESPONSE KRB_ERROR (81) message to the client.

To summarize, the sequence of steps for processing is as follows:

Servers:

1. For received KDC_REQ messages (with token ID 00 03)
   - process GSS framing (check OID)
     if the OID is not one of the two OID's specified in the GSSAPI Encapsulation section above, then process according to mechanism defined by that OID (if the OID is recognized). The processing is outside the scope of this specification. Otherwise, strip off GSS framing.
   - find KDC for specified realm (if KDC IP address cannot be obtained, send a KRB_ERROR message with error code KRB_IAKERB_ERR_KDC_NOT_FOUND to the client).
   - send to KDC (storing client IP address, port, and indication whether IAKERB proxy option or minimal messages option is being used)
   - retry with same or another KDC if no response is received. If the retries also fail, send an error message with error code KRB_IAKERB_ERR_KDC_NO_RESPONSE to the client.

2. For received KDC_REP messages
   - encapsulate with GSS framing, using token ID 01 03 and the OID that corresponds to the stored protocol option
   - send to client (using the stored client IP address and port)

3. For received AP_REQ and AP_REP messages
   - process locally per RFC 1964

Clients:

1. For sending KDC_REQ messages
   - create AS_REQ or TGS_REQ message
   - encapsulate with GSS framing (token ID 00 03 and OID corresponding to the protocol option).
   - send to server

2. For received KDC_REP messages
   - decapsulate by removing GSS framing (token ID 01 03)
   - process inner Kerberos message according to RFC 1510

3. For received AP_REQ and AP_REP messages
   - process locally per RFC 1964

7. The IAKERB minimal messages protocol

The client MAY initiate the IAKERB minimal messages variation when the number of messages must be minimized (the most significant reduction in the number of messages can occur when the client and the IAKERB proxy are in different realms). SPNEGO [8] may be used to securely negotiate between the protocols. A compliant IAKERB server MAY support the IAKERB minimal messages protocol.

(a) AS_REQ case: (used when the client does not have a TGT)

We extend the technique used in Hornstein [5]. The client indicates that the minimal message sub-protocol will be used by using the appropriate OID as described above. The client sends the GSS encapsulated AS_REQ message to the IAKERB proxy, and the IAKERB proxy processes the GSS framing (as described above for the IAKERB proxy option) and forwards the AS_REQ message to the KDC.

The IAKERB proxy will proxy the returned message (AS_REP or KRB_ERROR) from the KDC back to the client (after processing and removing the GSS framing). The protocol is complete in the KRB_ERROR case (from the server perspective, but the client should retry depending on the error type). In the AS_REP case, the IAKERB proxy will obtain the client's TGT from the AS_REP message before forwarding the AS_REP message to the client. The IAKERB proxy then sends a TGS_REQ message with the client's TGT in the additional tickets field to the client's KDC (ENC-TKT-IN-SKEY option).

The IAKERB proxy MAY handle returned KRB_ERROR messages and retry the TGS request message. Ultimately, the IAKERB proxy either proxies a KRB_ERROR message to the client, or it sends a GSS Initial Context token containing an AP_REQ message to the client. (Note: although the server sends the initial context token, the client is the initiator.) The IAKERB proxy MUST set the MUTUAL AUTH flag in the Initial Context token in order to cause the client to authenticate as well. The client will reply with the GSSAPI enscapsulated AP_REP message, if the IAKERB proxy's authentication succeeds. If all goes well, then, in order to enable subsequent efficient client authentications, the IAKERB proxy will then send a final message of type KRB_TKT_PUSH containing a Kerberos ticket (the reverse ticket) that has the IAKERB client principal identifier in the client identifier field of the ticket and its own principal identity in the server identifier field of the ticket:

```
KRB_TKT_PUSH :: = [APPLICATION 17] SEQUENCE {
  pvno[0]             INTEGER,  -- 5 (protocol version)
  msg-type[1]         INTEGER,  -- 17 (message type)
  ticket[2]           Ticket
}
```

The key used to encrypt the reverse ticket is a long term secret key chosen by the IAKERB proxy. The fields are identical to the AP_REQ ticket, except the client name will be switched with the server name, and the server realm will be switched with the client realm. (The one other exception is that addresses should not be copied unless the IAKERB proxy has included the client's address in the TGS_REQ message to the KDC). Sending the reverse ticket allows the client to efficiently initiate subsequent reauthentication attempts with a RFC1964 AP_REQ message. Note that the TKT_PUSH message is sent after mutual authentication and key establishment are complete.

(b) TGS_REQ case: (used when the client has a TGT)

The client indicates that the minimal messages sub-protocol will be used by using the appropriate OID as described above. The client initially sends a KRB_TKT_PUSH message (with the GSS header) to the IAKERB proxy in order to send it a TGT. The IAKERB proxy will obtain the client's TGT from the KRB_TKT_PUSH message and then proceed to send a TGS_REQ message to a KDC where the realm of the KDC is equal to the realm from the server realm field in the TGT sent by the client in the KRB_TKT_PUSH message. The protocol then continues as in the minimal messages AS_REQ case described above (see Figure 2); the IAKERB proxy's TGS_REQ message contains the client's TGT in the additional tickets field (ENC-TKT-IN-SKEY option). The IAKERB proxy then receives the TGS_REP message from the KDC and then sends a RFC 1964 AP_REQ message to the client (with the MUTUAL AUTH flag set - see AS_REQ case).

8. Addresses in Tickets

In IAKERB, the machine sending requests to the KDC is the server 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 RFC1510 [1], the addresses field of the AS and TGS requests SHOULD be blank and the caddr field of the ticket SHOULD similarly be left blank. One exception is in an AS request (where the request body is not integrity protected); the IAKERB proxy MAY add its own addresses and the addresses of the client to the AS request.

9. Combining IAKERB with Other Kerberos Extensions

This protocol is usable with other proposed Kerberos extensions such as PKINIT (Public Key Cryptography for Initial Authentication in Kerberos [4]). In such cases, the messages which would normally be sent to the KDC are instead sent by the client application to the server, which then forwards them to a KDC.

10. Security Considerations

In the minimal messages protocol option, the application server sends an AP_REQ message to the client. The ticket in the AP_REQ message SHOULD NOT contain authorization data since some operating systems may allow the client to impersonate the server and increase its own privileges. If the ticket from the server connotes any authorization, then the minimal messages protocol should not be used. Also, the minimal messages protocol may facilitate denial of service attacks in some environments; to prevent these attacks, it may make sense for the minimal messages protocol server to only accept a KRB_TGT_PUSH message on a local network interface (to ensure that the message was not sent from a remote malicious host).

11. Acknowledgements

We thank Ken Raeburn for his helpful comments.

12. References

[1] J. Kohl, C. Neuman, "The Kerberos Network Authentication Service (V5)", RFC 1510.

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

[3] J. Linn, "Generic Security Service Application Program Interface", RFC 2078.

[4] B. Tung, C. Neuman, M. Hur, A. Medvinsky, S. Medvinsky, J. Wray, J. Trostle, "Public Key Cryptography for Initial Authentication in Kerberos", WORK IN PROGRESS Internet Draft draft-ietf-cat-kerberos-pkinit-12.txt.

[5] K. Hornstein, T. Lemon, B. Aboba, J. Trostle, "DHCP Authentication via Kerberos V", WORK IN PROGRESS Internet Draft draft-hornstein-dhc-kerbauth-02.txt.

[6] S. Bradner, "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

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

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

12. Author's Addresses

Michael Swift
University of Washington
Seattle, WA
Email: mikesw@cs.washington.edu

Jonathan Trostle
Cisco Systems
170 W. Tasman Dr.
San Jose, CA 95134, U.S.A.
Email: jtrostle@cisco.com
Phone: (408) 527-6201

Bernard Aboba
Microsoft
One Microsoft Way
Redmond, Washington, 98052, U.S.A.
Email: bernarda@microsoft.com

Glen Zorn
Cisco Systems
Bellevue, WA U.S.A.
Email: gwz@cisco.com
Phone: (425) 468-0955

This draft expires on January 31st, 2002.