Network Working Group
INTERNET-DRAFT
Category: Standards Track

Jonathan Trostle
Cisco Systems
Mike Swift
University of WA
John Brezak
Microsoft
Bill Gossman
Cisco Systems

Kerberos Set/Change Password: Version 2
<draft-ietf-cat-kerberos-set-passwd-06.txt>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026 [6].

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This draft expires on December 31st, 2001. Please send comments to the authors.

1. Abstract

This proposal specifies a Kerberos (RFC 1510 [3]) change/set password protocol and a Kerberos change/set key protocol. The protocol consists of a single request and reply message. The request message includes both AP-REQ and KRB-PRIV submessages; the new password is contained in the KRB-PRIV submessage which is encrypted in the subsession key from the AP-REQ. The original Kerberos change password protocol did not allow for an administrator to set a password for a new user. This functionality is useful in some environments, and this proposal allows password setting as well as password changing. The protocol includes fields in the request message to indicate the

principal which is having its password set. We also extend the set/change protocol to allow a client to send a sequence of keys to

the KDC instead of a cleartext password. If in the cleartext password case, the cleartext password fails to satisfy password policy, the server should use the result code KRB5_KPASSWD_POLICY_REJECT.

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. Definitions from RFC 1510

We include some of the relevant ASN.1 definitions from RFC 1510 in this section.

```
Realm ::=           GeneralString

PrincipalName ::=   SEQUENCE {
                    name-type[0]     INTEGER,
                    name-string[1]   SEQUENCE OF GeneralString
}

KerberosTime ::=    GeneralizedTime
                    -- Specifying UTC time zone (Z)

HostAddress ::=     SEQUENCE  {
                    addr-type[0]             INTEGER,
                    address[1]               OCTET STRING
}

EncryptedData ::=   SEQUENCE {
                  etype[0]     INTEGER, -- EncryptionType
                  kvno[1]      INTEGER OPTIONAL,
                  cipher[2]    OCTET STRING -- ciphertext
}

EncryptionKey ::=   SEQUENCE {
                    keytype[0]    INTEGER,
                    keyvalue[1]   OCTET STRING
}

Checksum ::=        SEQUENCE {
                    cksumtype[0]   INTEGER,
                    checksum[1]    OCTET STRING
}


AP-REQ ::= [APPLICATION 14] SEQUENCE {
        pvno [0]         INTEGER,        -- indicates Version 5
```

```
            msg-type [1]     INTEGER,        -- indicates KRB_AP_REQ
            ap-options[2]    APOptions,
            ticket[3]        Ticket,
            authenticator[4] EncryptedData
      }
```

```
      APOptions ::= BIT STRING {

              reserved (0),
              use-session-key (1),
              mutual-required (2)
      }

      Ticket ::= [APPLICATION 1] SEQUENCE {
              tkt-vno [0]     INTEGER,        -- indicates Version 5
              realm [1]       Realm,
              sname [2]       PrincipalName,
              enc-part [3]    EncryptedData
      }


      -- Encrypted part of ticket
      EncTicketPart ::= [APPLICATION 3] SEQUENCE {
              flags[0]        TicketFlags,
              key[1]          EncryptionKey,
              crealm[2]       Realm,
              cname[3]        PrincipalName,
              transited[4]    TransitedEncoding,
              authtime[5]     KerberosTime,
              starttime[6]    KerberosTime OPTIONAL,
              endtime[7]      KerberosTime,
              renew-till[8]   KerberosTime OPTIONAL,
              caddr[9]        HostAddresses OPTIONAL,
              authorization-data[10]  AuthorizationData OPTIONAL
      }

      -- Unencrypted authenticator
      Authenticator ::= [APPLICATION 2] SEQUENCE  {
              authenticator-vno[0]    INTEGER,
              crealm[1]               Realm,
              cname[2]                PrincipalName,
              cksum[3]                Checksum OPTIONAL,
              cusec[4]                INTEGER,
              ctime[5]                KerberosTime,
              subkey[6]               EncryptionKey OPTIONAL,
              seq-number[7]           INTEGER OPTIONAL,
              authorization-data[8]   AuthorizationData OPTIONAL
      }

      AP-REP ::= [APPLICATION 15] SEQUENCE {
              pvno [0]        INTEGER,        -- represents Kerberos V5
              msg-type [1]    INTEGER,        -- represents KRB_AP_REP
              enc-part [2]    EncryptedData
      }
```

```
EncAPRepPart ::= [APPLICATION 27] SEQUENCE {
        ctime [0]       KerberosTime,
        cusec [1]       INTEGER,
        subkey [2]      EncryptionKey OPTIONAL,
        seq-number [3]  INTEGER OPTIONAL
```

```
   }
```

Here is the syntax of the KRB-ERROR message:

```
   KRB-ERROR ::=   [APPLICATION 30] SEQUENCE {
           pvno[0]         INTEGER,
           msg-type[1]     INTEGER,
           ctime[2]        KerberosTime OPTIONAL,
           cusec[3]        INTEGER OPTIONAL,
           stime[4]        KerberosTime,
           susec[5]        INTEGER,
           error-code[6]   INTEGER,
           crealm[7]       Realm OPTIONAL,
           cname[8]        PrincipalName OPTIONAL,
           realm[9]        Realm, -- Correct realm
           sname[10]       PrincipalName, -- Correct name
           e-text[11]      GeneralString OPTIONAL,
           e-data[12]      OCTET STRING OPTIONAL
   }
```

The KRB-PRIV message is used to send the request and reply data:

```
   KRB-PRIV ::=     [APPLICATION 21] SEQUENCE {
                pvno[0]                 INTEGER,
                msg-type[1]             INTEGER,
                enc-part[3]             EncryptedData
   }

   EncKrbPrivPart ::=   [APPLICATION 28] SEQUENCE {
                user-data[0]            OCTET STRING,
                timestamp[1]            KerberosTime OPTIONAL,
                usec[2]                 INTEGER OPTIONAL,
                seq-number[3]           INTEGER OPTIONAL,
                s-address[4]            HostAddress,
                                                -- sender's addr
                r-address[5]            HostAddress OPTIONAL
                                                -- recip's addr
   }
```

4. The Protocol

The service SHOULD accept requests on UDP port 464 and TCP port 464 as well. Use of other ports can significantly increase the complexity and size of IPSEC policy rulesets in organizations that have IPSEC capable nodes.

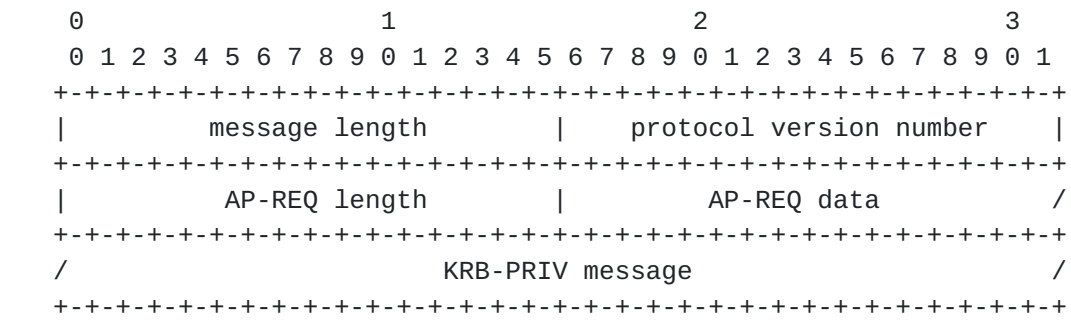
The protocol consists of a single request message followed by a single reply message. For UDP transport, each message must be fully

contained in a single UDP packet.

For TCP transport, there is a 4 octet header in network byte order that precedes the message and specifies the length of the message. This requirement is consistent with the TCP transport header in

1510bis.

Request Message



All 16 bit fields are in network byte order.

message length field: contains the number of bytes in the message including this field.

protocol version number: contains the hex constant 0x0002 (network byte order).

AP-REQ length: length of AP-REQ data, in bytes. If the length is zero, then the last field contains a KRB-ERROR message instead of a KRB-PRIV message.

AP-REQ data: (see [3]) For a change password/key request, the AP-REQ message service ticket sname, srealm principal identifier is kadmin/changepw@REALM where REALM is the realm of the change password service. The same applies to a set password/key request except the principal identifier is kadmin/setpw@REALM. The authenticator in the AP-REQ MUST contain a subsession key (which will be used to encrypt the KRB-PRIV user data field - see below). The KDC may have stronger pseudorandom generating capability then the clients; thus, the client SHOULD use the session key as an input (along with additional locally pseudorandom generated bits) into the generation of the subsession key. To enable setting of passwords/keys, it is not required that the initial flag be set in the Kerberos service ticket. The initial flag is required for change requests, but not for set requests. We have the following definitions:

	old passwd in request?	initial flag required?	target principal can be distinct from authenticating principal?
change password:	yes	yes	no

set password:	no	policy (*)	yes
set key:	no	policy (*)	yes
change key:	no	yes	no

policy (*): implementations SHOULD allow administrators to set the initial flag required for set requests policy to either yes or no. Clients MUST be able to retry set requests that fail due to error 7 (initial flag required) with an initial ticket. Clients SHOULD NOT cache service tickets targetted at kadmin/changepw.

KRB-PRIV message (see [3]) This KRB-PRIV message must be encrypted using the subsession key from the authenticator in the AP-REQ. The authenticator MUST contain a subsession key. The timestamp and usec fields of the KRB-PRIV message MUST be present, and the data values MUST be copies of the same data values from the authenticator. The recipient should ignore the sender address field in the KRB-PRIV message.

The user-data component of the message contains the DER encoding of the ChangePasswdData ASN.1 type described below:

```
 ChangePasswdData ::=  SEQUENCE {
                     passwds [0]          PasswordSequence OPTIONAL,
                     keys [1]             KeySequences OPTIONAL,
                       -- exactly one of the above two will be
                       -- present, else KRB5_KPASSWD_MALFORMED
                       -- error will be returned by the server.
                     targname[2]          PrincipalName OPTIONAL,
                       -- only present in set password/key: the
                       -- principal which will have its password
                       -- or keys set. Not present in a set request
                       -- if the client principal from the ticket is
                       -- the principal having its passwords or keys
                       -- set.
                     targrealm[3]         Realm OPTIONAL,
                       -- only present in set password/key: the realm
                       -- for the principal which will have its
                       -- password or keys set. Not present in a set
                       -- request if the client principal from the
                       -- ticket is the principal having its
                       -- passwords or keys set.
                     flags[4]             RequestFlags OPTIONAL
                       -- 32 bit string
                     }

 KeySequences ::= SEQUENCE (SIZE (1..MAX)) OF KeySequence

 KeySequence ::= SEQUENCE {
                   key[0]        EncryptionKey,
                   salt[1]       OCTET STRING OPTIONAL,
                         -- depends on enc type, not currently used
                   salt-type[2]  INTEGER OPTIONAL
```

```
                        -- depends on enc type, not currently used
    }

    PasswordSequence ::= SEQUENCE {
                         newpasswd[0]  OCTET STRING,
                         oldpasswd[1]  OCTET STRING OPTIONAL
```

```
                        -- oldpasswd always present for change
                        -- password but not present for set
                        -- password, set key, or change key
                        -- NOTE: the passwords are UTF8 strings.
    }

    RequestFlags ::= BIT STRING (SIZE (32..MAX))
                      -- reserved(0)
                      -- request-srv-gen-keys(1)
                           -- only in change/set keys
                           -- if the client desires
                           -- server to contribute to
                           -- keys;
                           -- server will return keys
```

The server must verify the AP-REQ message, check whether the client principal in the ticket is authorized to set/change the password/keys (either for that principal, or for the principal in the targname field if present), and decrypt the new password/keys. The server also checks whether the initial flag is required for this request, replying with status 0x0007 if it is not set and should be. An authorization failure is cause to respond with status 0x0005. For forward compatibility, the server should be prepared to ignore fields after targrealm in the structure that it does not understand.

If the passwds field is present, it contains the new cleartext password (with the old cleartext password for a change password operation). Otherwise the keys field is present, and it contains a sequence of encryption keys.

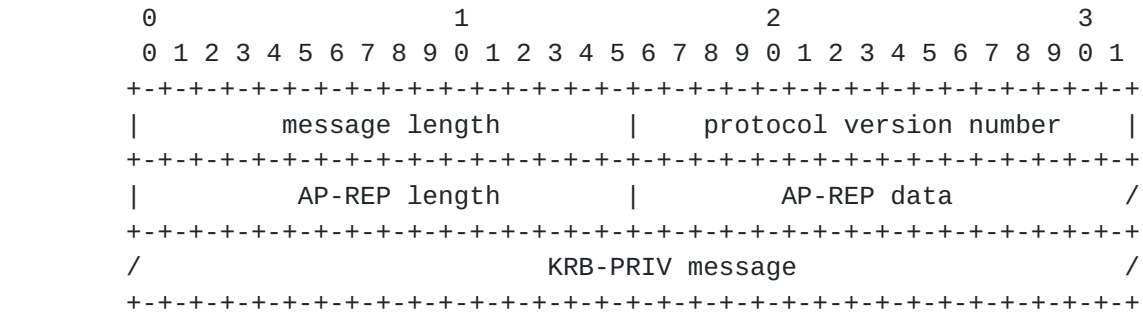
In the cleartext password case, if the old password is sent in the request, the request MUST be a change password request. If the old password is not present in the request, the request MUST be a set password request. The server should apply policy checks to the old and new password after verifying that the old password is valid. The server can check validity by obtaining a key from the old password with a keytype that is present in the KDC database for the user and comparing the keys for equality. The server then generates the appropriate keytypes from the password and stores them in the KDC database. If all goes well, status 0x0000 is returned to the client in the reply message (see below). For a change password operation, the initial flag in the service ticket MUST be set.

In the key sequence case, the sequence of keys is sent to the change or set password service (kadmin/changepw or kadmin/setpw respectively). For a principal that can act as a server, its preferred keytype should be sent as the first key in the sequence,

but the KDC is not required to honor this preference. Application servers SHOULD use the key sequence option for changing/setting their keys. The change/set password services should check that all keys are in the proper format, returning the KRB5_KPASSWD_MALFORMED error otherwise.

For change/set key, the request message may include the request flags bit string with the request-srv-gen-keys bit set. In this case, the client is requesting that the server add entropy to its keys in the KeySequences field. When using this option, the client SHOULD attempt to generate pseudorandom keys with as much entropy as possible in its request. The server will return the final key sequence in a KeySequences structure in the edata of the reply message. The server does not store any of the new keys at this point. The client MUST make a subsequent change/set key request without the request-srv-gen-keys bit; if the server returns KRB5_KPASSWD_SUCCESS for this second request, then the new keys have been written into the database. A conformant server MUST support this option.

Reply Message



All 16 bit fields are in network byte order.

message length field: contains the number of bytes in the message including this field.

protocol version number: contains the hex constant 0x0002 (network byte order). (The reply message has the same format as in the original Kerberos change password protocol).

AP-REP length: length of AP-REP data, in bytes. If the length is zero, then the last field contains a KRB-ERROR message instead of a KRB-PRIV message. An implementation should check this field to determine whether a KRB-ERROR message or KRB-PRIV message has been returned.

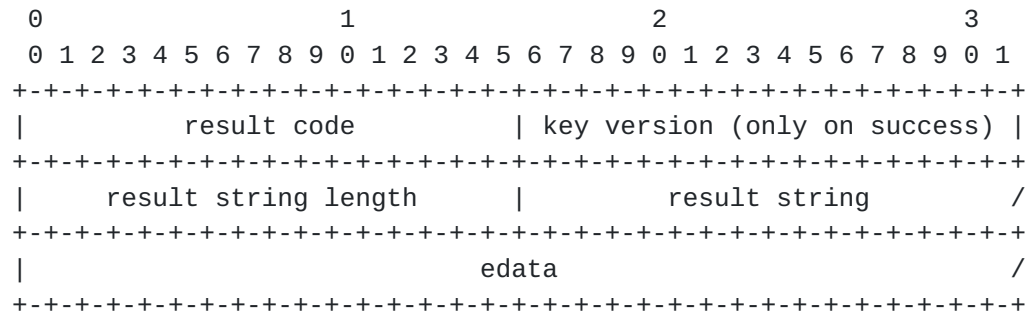
AP-REP data: the AP-REP is the response to the AP-REQ in the request packet. The subkey MUST be present in the AP-REP message.

KRB-PRIV message: This KRB-PRIV message must be encrypted using the subkey from the AP-REP message. The client should ignore the sender address (the server's address) in the KRB-PRIV message. Reflection attacks are prevented since the subkey is used to encrypt the user-data field of the KRB-PRIV message. The timestamp and usec fields of

the KRB-PRIV message MUST be present, and the data values MUST be copies of the same data values from the AP-REP message.

The server will respond with a KRB-PRIV message unless it cannot validate the client AP-REQ or KRB-PRIV message, in which case it will respond with a KRB-ERROR message.

The user-data component of the KRB-PRIV message, or e-data component of the KRB-ERROR message, must consist of the following data.



result code (16 bits) (result codes 0-4 are the same as in the original Kerberos change password protocol):

The result code must have one of the following values (network byte order):

KRB5_KPASSWD_SUCCESS	0	request succeeds (This value is not allowed in a KRB-ERROR message)
KRB5_KPASSWD_MALFORMED	1	request fails due to being malformed
KRB5_KPASSWD_HARDERROR	2	request fails due to "hard" error in processing the request (for example, there is a resource or other problem causing the request to fail)
KRB5_KPASSWD_AUTHERROR	3	request fails due to an error in authentication processing
KRB5_KPASSWD_SOFTERROR	4	request fails due to a soft error in processing the request
KRB5_KPASSWD_ACCESSDENIED	5	requestor not authorized

KRB5_KPASSWD_BAD_VERSION 6 protocol version unsupported

KRB5_KPASSWD_INITIALFLAG_NEEDED 7 initial flag required

KRB5_KPASSWD_POLICY_REJECT 8 new cleartext password fails policy; the result string should include a text message to be presented to the user.

KRB5_KPASSWD_WRONG_SRV 9 policy failure: the client sent change/set key and should have sent change/set passwd, or vice-versa.

KRB5_KPASSWD_BAD_PRINCIPAL 10 target principal does not exist (only in response to a set password or set key request).

KRB5_KPASSWD_ETYPE_NOSUPP 11 the request contains a key sequence containing at least one etype that is not supported by the KDC. The response edata contains an ASN.1 DER encoded PKERB-ETYPE-INFO type that specifies the etypes that the KDC supports:

```
KERB-ETYPE-INFO-ENTRY :: = SEQUENCE
{
     encryption-type[0]  INTEGER,
     salt[1]        OCTET STRING
       OPTIONAL -- not sent, client
                -- may ignore if
                -- sent
}

PKERB-ETYPE-INFO ::= SEQUENCE OF
                KERB-ETYPE-INFO-ENTRY
```

The client should retry the request using only etypes (keytypes) that are contained within the PKERB-ETYPE-INFO structure in the previous response.

KRB5_KPASSWD_ETYPE_SRVGENKEYS 12 See the following paragraph.

The KRB5_KPASSWD_ETYPE_SRVGENKEYS result code is returned when the request has the request-srv-gen-keys flag set and the server is returning the KeySequences structure defined above in the edata field of the reply. The server returns one key sequence

structure of the same keytype for each key sequence structure in the client request, unless it does not support one of the keytypes (or etypes). In that case, it returns error KRB5_KPASSWD_ETYPE_NOSUPP as discussed above. The server MUST add keylength number of bits of entropy to each key, where

keylength is the number of actual key bits in the key (minus any parity or non-entropy contributing bits). The assumption here is that the client may have added insufficient entropy to the request keys. The server SHOULD use the client key from each KeySequence structure as input into the final keyvalue for the returned key. The client MUST make another request after receiving a reply with this status, since no keys have been written into the database.

0xFFFF is returned if the request fails for some other reason. The client must interpret any non-zero result code as a failure.

key version (16 bits - optional):
Present if and only if the result code is KRB5_KPASSWD_SUCCESS. This contains the key version of the new key(s).

result string length (16 bits):
Gives the length of the following result string field, in bytes. If the result string is not present, the length is zero.

result string (optional):
This field is a UTF-8 encoded string which can be displayed to the user. Specific reasons for a password set/change policy failure is one possible use for this string.

edata (optional):
Used to convey additional information as defined by the result code.

5. Acknowledgements

The authors thank Ken Raeburn, Tom Yu, Martin Rex, Sam Hartman, Tony Andrea, Nicolas Williams, and other participants from the IETF Kerberos Working Group for their input to the document.

6. Security Considerations

Password policies should be enforced to make sure that users do not pick passwords (for change password/key) that are vulnerable to brute force password guessing attacks.

7. References

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

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

[3] J. Kohl, C. Neuman. The Kerberos Network Authentication
Service (V5), Request for Comments 1510.

8. Expiration Date

This draft expires on December 31st, 2001.

9. Authors' Addresses

Jonathan Trostle
Cisco Systems
170 W. Tasman Dr.
San Jose, CA 95134
Email: jtrostle@cisco.com

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

John Brezak
Microsoft
1 Microsoft Way
Redmond, WA 98052
Email: jbrezak@microsoft.com

Bill Gossman
Cisco Systems
500 108th Ave. NE, Suite 500
Bellevue, WA 98004
Email: bgossman@cisco.com

10. Full Copyright Statement

HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."