INTERNET-DRAFT
draft-ietf-cat-kerberos-set-passwd-04.txt
March 2001

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

Kerberos Set/Change Password: Version 2

0. Status Of This Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026 [1].

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.

Comments and suggestions on this document are encouraged. Comments on this document should be sent to the CAT working group discussion list: ietf-cat-wg@stanford.edu.

This draft expires on September 30th, 2001.

1. Abstract

The Kerberos (RFC 1510 [3]) change password protocol (Horowitz [4]), does not allow for an administrator to set a password for a new user. This functionality is useful in some environments, and this proposal extends [4] to allow password setting. The changes are: adding new fields to the request message to indicate the principal which is having its password set, not requiring the initial flag in the service ticket, using a new protocol version number, and adding three new result codes. 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

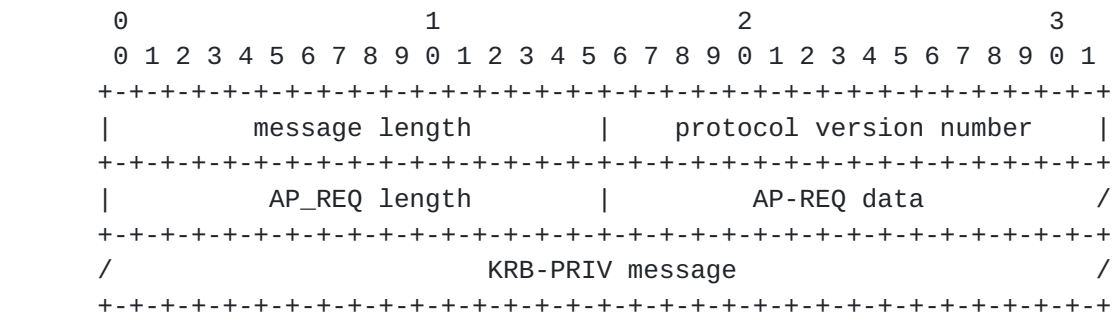
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [2].

3. The Protocol

The service MUST accept requests on UDP port 464 and TCP port 464 as well. 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 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. 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 session key from the ticket in the AP-REQ.

The user-data component of the message consists of the following ASN.1 encoded structure:

```
ChangePasswdData :: =  SEQUENCE {
                    newpasswdorkeys[0]   NewPasswdOrKeys,
                    targname[1]          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[2]         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[3]             RequestFlags OPTIONAL
                      -- 32 bit string
                    }

NewPasswdOrKeys :: = CHOICE {
                    passwords[0]  PasswordSequence,  -- change/set passwd
                    keyseq[1]     KeySequences       -- change/set key
}

KeySequences :: = SEQUENCE OF KeySequence

KeySequence :: = SEQUENCE {
                    key[0]        EncryptionKey,
```

```
                    salt[1]       OCTET STRING OPTIONAL,
                    salt-type[2]  INTEGER OPTIONAL
}

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
}

RequestFlags :: = BIT STRING {
                  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 or change the password (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.

The newpasswdorkeys field contains either the new cleartext password (with the old cleartext password for a change password operation), or a sequence of encryption keys with their respective salts.

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.

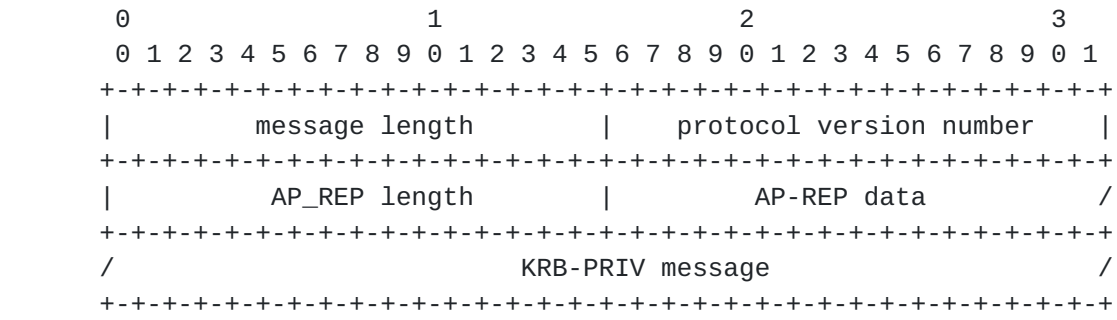
In the key sequence case, the sequence of keys is sent to the change or set password service (kadmin/changepw or kadmin/setpw respectively). For a principal that can act as a server, its preferred keytype should be sent as the first key in the sequence, but the KDC is not required

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 [4]).

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.

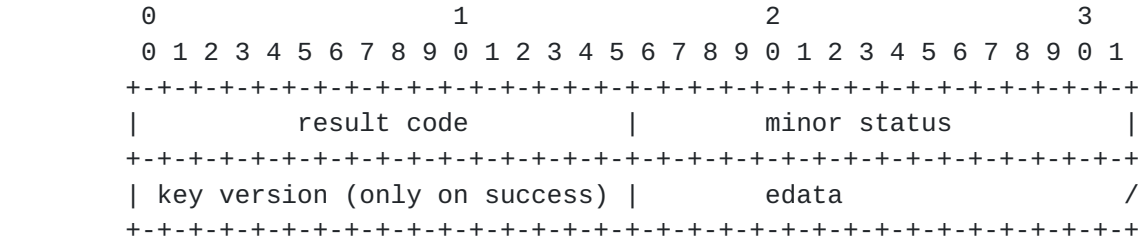
AP-REP data: the AP-REP is the response to the AP-REQ in the request packet.

KRB-PRIV from [4]: This KRB-PRIV message must be encrypted using the session key from the ticket in the AP-REQ.

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 from [4]):
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_BAD_PRINCIPAL	9	target principal does not exist (only in response to a set password or set key request).
KRB5_KPASSWD_ETYPE_NOSUPP	10	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 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
                  -- ignores 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 11 the request has the request-srv-gen-keys flag set and the server is returning the KeySequence 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. 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.

0xFFFF is returned if the request fails for some other reason.

The client must interpret any non-zero result code as a failure.

minor status (16 bits):

This field contains a minor status code. It can be used to index into a catalog of strings and the indexed string can then be

displayed to the user. Additional information on a password set/change policy failure is one use for this string.

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).

edata: used to convey additional information as defined by the result code.

4. Acknowledgements

The authors thank Tony Andrea for his input to the document.

5. 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.

[4] M. Horowitz. Kerberos Change Password Protocol, ftp://ds.internic.net/internet-drafts/draft-ietf-cat-kerb-chg-password-02.txt

6. Expiration Date

This draft expires on September 30th, 2001.

7. 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
1 Microsoft Way
Redmond, WA 98052
Email: jbrezak@microsoft.com

Bill Gossman
Cybersafe Corporation
1605 NW Sammamish Rd.

Issaquah, WA 98027-5378
Email: bill.gossman@cybersafe.com