Kerberos Working Group   Jonathan Trostle
INTERNET-DRAFT   Cisco Systems
Category: Standards Track   Mike Swift
                                                                University of WA
                                                                    John Brezak
                                                                      Microsoft
                                                                   Bill Gossman
                                                                  Cisco Systems
                                                               Nicolas Williams
                                                               Sun Microsystems

Kerberos Set/Change Password: Version 2
<draft-ietf-krb-wg-kerberos-set-passwd-00.txt>

Status of this Memo

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

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.

Copyright Notice

Abstract

This proposal specifies an extensible protocol for setting keys and changing the passwords of Kerberos [RFC1510] principals.

The protocol support a single operation per-session when run over UDP, or multiple operations per-session when run over TCP. Clients can change their own principal's password or keys or they can change other principals', provided that they are properly authorized to do so.

Additional related features include the ability to determine the known aliases of Kerberos principals. This feature will facilitate the implementation of aliasing of target principal names in KDC requests by allowing principals to know which names are aliases of their canonical principal names. Principal aliasing is needed to properly support the use of aliases and short-form names by users without requiring that clients canonicalize principal names, possibly using insecure name services in the process.

This protocol uses IETF language tags [RFC3066] to negotiate proper localization of help messages intended for users. UTF-8 is used throughout for strings, suitably constrained, where necessary, by the minor version of Kerberos V in use by clients and servers.

1. Introduction

Kerberos lacks a single, standard protocol for changing passwords and keys. While several vendor-specific protocols exist for changing Kerberos passwords/keys, none are properly internationalized.

This document defines a protocol that is somewhat backward-compatible with the "kpasswd" protocol, version 1 [KPASSWDv1] and a derivative defined in [RFC3244] that uses more or less the same protocol framing.

This new protocol is designed to be extensible and properly internationalized.

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. Table of Contents

4. The Protocol

The structure of the protocol is quite similar to that of typical RPC protocols. Each operation has a structure for each client request and a structure for each server response. Each transaction consists of a single operation; the abstract syntax for the protocol implies the use, on the wire, of an operation identifier associated with an opaque blob representing the request of response. The protocol data is wrapped in a KRB-PRIV and framed in a header that is backwards compatible with version 1 of this protocol.

4.1 Transports

The service SHOULD accept requests on UDP port 464 and TCP port 464. This is the same port used by version 1 [KPASSWDv1] of this protocol, but version 2 is a completely different protocol sharing with version 1 only the outer framing.

4.2 Protocol Framing

For compatibility with the original Kerberos password changing protocol developed at MIT, the first 4 bytes of the message consist of a 2-byte network byte order message length, followed by a 2 byte network byte order protocol version number, followed by a 2 byte network byte order length for an optional AP-REQ, AP-REP or KRB-ERROR, followed by the same, if present, followed by a KRB-PRIV (optional in TCP) containing the actual protocol message encoded in DER [X690].

In the case of TCP there is an additional 4 byte network byte order length prepended to the frame described above.

The protocol version number MUST be set to 2 for this protocol.

Bytes on the wire description of the framing:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         message length        |    protocol version number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| AP-REQ length (0 if absent)   | AP-REQ data (if present)      /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                    KRB-PRIV message                           /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The same framing applies equally to requests and responses, but responses use AP-REP and/or KRB-ERROR instead of AP-REQ:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         message length        |    protocol version number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| AP-REP length (0 if absent)   | AP-REP data (if present)      /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                    KRB-PRIV message                           /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         message length        |    protocol version number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| KRB-ERROR length (0 if absent)| KRB-ERROR data (if present)   /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

For the UDP case the AP-REQ/AP-REP/KRB-ERROR MUST always be included.

Note that this framing is used by version 1 [KPASSWDv1] and version 0xff80 [RFC3244], though the latter does not use the framing when responding with KRB-ERROR messages.

Servers MAY respond to version 0xff80 requests with an un-framed

KRB-ERROR and e-data set as per-RFC3244 [RFC3244], otherwise clients and server MUST always use this framing. See section 4.3.

4.2.1 The protocol over UDP

In the UDP case there is a single message from the client and a single response from the server with no state kept between requests, and each request MUST include a Kerberos AP-REQ and a KRB-PRIV and each response MUST carry an AP-REP, or KRB-ERROR and a KDB-PRIV. Both the client and server MUST destroy the authentication context after each operation.

UDP clients MUST not request the use of sequence numbers, otherwise it cannot generate the KRB-PRIV prior to receiving the AP-REP. Clients MAY refuse to operate version 2 of the protocol over UDP; it is RECOMMENDED that servers reject version 2 UDP requests.

4.2.2 The protocol over TCP

When used with the TCP transport, there is a 4 octet header in network byte order that precedes the message and specifies the length of the message.

The initial message from the client MUST carry an AP-REQ and the response to any request bearing an AP-REQ MUST carry an AP-REP.

Subsequent messages MAY involve Kerberos V AP exchanges, but generally the client SHOULD NOT initiate a new AP exchange except when it desires to authenticate as a different principal, when its current authentication context is about to expire or when the server responds with an error indicating that the client must re-initialize the authentication context (possibly due to the previous context expiring).

The server MUST NOT process any requests that do not contain an AP-REQ unless a non-expired authentication context is currently established with the client on the same TCP connection.

Servers MAY close open sessions at any time.

4.3 Protocol version negotiation

There are several major versions of this protocol. Version 2 also introduces a notion of protocol minor versions for use in negotiating protocol extensions. As of this time only one minor version is defined for major version 2: minor version 0.

4.3.1 Protocol major version negotiation

Version 2 clients that also support other versions, such as [KPASSWDv1] or [rfc3244] SHOULD attempt to use version 2 of the protocol first and then try other versions if the server responds with either a message framed as described in section 4.2 but

with a protocol version number other than 2 (in the case of
[KPASSWDv1], or a KRB-ERRROR with an error code of
KRB5_KPASSWD_BAD_VERSION in the e-data field [RFC3244].

Note that some version 1 servers return a KRB-ERROR indicating that versions other than 1 of the change password protocol are not supported rather than an AP-REP and a KRB-PRIV containing the error data. Therefore change password protocol negotiation is subject to downgrade attacks where version 2 clients support version 1 of this protocol.

Also note that some [RFC3244] implementations do not return any responses to requests for protocol versions other than 0xff80, and in the TCP case close the TCP connection.

Version 2 servers MAY support other versions of the Kerberos password change protocol.

Version 2 servers SHOULD respond to non-v2 requests using whatever response is appropriate for the versions used by the clients, but if a server does not do this or know how to do this then it MUST respond with an error framed as in section 4.2, using an AP-REP and KRB-PRIV if the client's AP-REQ can be accepted, or a KRB-ERROR (framed) otherwise and using a ProtocolErrorCode value of unsupported-major-version.

4.3.2 Protocol minor version negotiation

Version 2 clients are free to use whatever protocol minor version and message extensions are available to them in their initial messages to version 2 servers, provided that the minor versions (other than 0) have been defined through IETF documents and registered with the IANA.

Version 2 clients and servers MUST support all protocol minor versions between 0 to the highest version supported by the client and server. That is, a client or server that supports minor version 4 MUST also support minor versions 0, 1, 2 and 3.

Version 2 servers MUST answer with the highest protocol minor version number supported by the server and the client.

Version 2 clients MUST use the protocol minor version used in a server's reply for any subsequent messages in the same session (currently this only applies to TCP sessions).

See section 4.7 for further description of the protocol's extensibility and its relation to protocol minor versions and the negotiation thereof.

4.4 Use of Kerberos V

This protocol makes use of messages defined in [RFC1510] and [clarifications]. Specifically, AP-REQ, AP-REP, KRB-ERROR and KRB-PRIV. Because of the proposed extensions to Kerberos V which will require a new ASN.1 module, and because of the ways that the

Kerberos V ASN.1 types will change, this protocol cannot safely import any types from the Kerberos V module, therefore the Kerberos PDUs are encoded as OCTET STRINGs herein.

All operations are to be performed by the server on behalf of the client principal.

The client SHOULD use "kadmin/changepw" as the server principal name for this protocol. The server MUST have a principal name of "kadmin/changepw" and MAY have a principal name of "kadmin/setpw."

The client MUST request mutual authentication and the client MUST NOT request the use of sequence numbers when using the protocol over UDP, but it MUST request the use of sequence numbers when running over TCP.

The server MUST reject requests that operate on the same principal as the client's if the client's principal is not in the same realm as the server's principal name or if the client's ticket is not INITIAL.

The server MAY reject all requests from clients operating on principals not in the client's realm. The server MAY reject all requests operating on principals other than the client's.

4.4.1 Use of KRB-ERROR

When an error arises during the AP exchange for which [clarifications] does not provide an appropriate error code then the server MUST use KRB_ERR_GENERIC as the error, a localized (if possible [er, is that ok, pre-extensions? probably not]) error string for the e-text field of KRB-ERROR and the encoding of an Error-Response PDU (see section 6) as e-data.

4.5 Use of ASN.1

This protocol's messages are defined in ASN.1, using only features from [X680]. All ASN.1 types defined herein are to be encoded in DER [X690]. A complete ASN.1 module is given in section 6. The ASN.1 tagging environment for this module is EXPLICIT.

The DER encoding of the ASN.1 PDUs are exchanged wrapped in a KRB-PRIV as described above.

4.6 Protocol internationalization

Protocol requests have an optional field indicationg the languages spoken by the client user; the client SHOULD send its list of spoken languages to the server (once per-TCP session), but if future extensions to the Kerberos protocol should add similar functionality then the client SHOULD NOT use this field when using the extended

Kerberos protocol. All strings in the protocol are UTF-8 strings. The server SHOULD localize all strings intended for users to a language in common with the languages spoken by the client user.

For TCP sessions servers MUST cache the optional language tag lists from prior requests for use with requests that exclude the language tag list. Clients MAY expect such server behaviour and send the language tag lists only when they change or even just once per-TCP session. Clients SHOULD send the server the language tag list at least once, with or before any actual operation.

Kerberos principal and realm names used in this protocol MUST be constrained as per the specification of the version of Kerberos V used by the client.

4.6.1 Normalization forms for UTF-8 strings

No normalization form is required for string types other than for PrincipalName and Realm, which two types are constrained by the specification of the version of Kerberos V used by the client, and the password fields in the change password operation, which MUST be normalized according to [k5stringprep].

4.6.2 Language negotiation

The server MUST pick a language from the client's input list or the default language tag (see [RFC3066]) for text in its responses which is meant for the user to read.

The server SHOULD use a language selection algorithm such that consideration is first given to exact matches between the client's spoken languages and the server's available locales, followed by "fuzzy" matches where only the first sub-tags of the client's language tag list are used for matching against the servers available locales.

When the server has a message catalog for one of the client's spoken languages the server SHOULD localize any text strings intended for users to read.

4.7 Protocol Extensibility

The protocol is defined in ASN.1 and uses extensibility markers throughout. As such, the module presented herein can be extended within the framework of [X680].

Typed holes are not used in this protocol as it is very simple and does not require the ability to deal with abstract data types defined in different layers. For this reason, the only way to extend this protocol is by extending the ASN.1 module within the framework of the IETF; all future extensions to this protocol have to be defined in IETF documents unless otherwise specified in a future IETF revision of this protocol.

A protocol minor version number is used to negotiate use of extensions. See section 4.3.2 for the minor version negotiation.

Message extensions are to be closely tied to protocol minor numbers. Clients MAY use any protocol minor version that they support in initial requests, and MUST use the protocol minor version indicated in the server's initial reply in any subsequent requests in the same session (this only applies in the TCP case). Clients MAY cache the minor version number supported by any given server for a reasonably short and finite amount of time - 24 hours is the maximum RECOMMENDED time for caching server minor version information.

Servers SHOULD ignore protocol extensions and minor versions that they do not understand in initial requests, except for extensions to the "Op-req" type, which MUST result in an error; servers MAY respond with an error (ProtocolErrorCode value of unsupported-minor-version) to clients that use minor versions unsupported by the server in their initial requests.

Servers MUST select the highest minor version in common with their clients for use in replies.

Servers MAY support a subset of the operations defined in this protocol but MUST support all the PDUs.

4.8 Protocol Subsets

The structure of the protocol is such that the ASN.1 syntaxes for the various operations supported by the protocol are independent of the each other. Clients and servers MAY implement subsets of the overall protocol.

The structure of this protocol and the properties of the tag-length-value (TLV) DER encoding of ASN.1 make it possible to describe the encoding of individual operations' messages very simply.

In the interest of facilitating ease of implementation for trivial subsets of this protocol, without the need for ASN.1 compilers, section 10 describes examples of TLV layouts of some individual protocol operations (but the DER encodings of tags, lengths and UNIVERSAL values is not described).

5 Protocol Operations

The protocol as defined herein supports the following operations relating to the management of Kerberos principal's passwords or keys:

- change password
- set key
- get password policy name and/or description of principal
- list aliases of a principal

- list enctypes supported by realm

These operations are needed to support Kerberos V interoperability between clients and KDCs of different implementation origins.

The operation for retrieving a list of aliases of a principal is needed where KDCs implement aliasing of principal names and allows clients to properly setup their "keytabs" when principal aliasing is in use.

Operations such as creation or deletion of principals are outside the scope of this document, and should be performed via directories or other Kerberos administration protocols. However, it is conceivable that such operations could be added to this protocol at a later point.

Operations can be added to the protocol only via future IETF RFCs.

The individual operations are described in section 7.

5.1 PDUs

The types "Request," "Response" and "Error-Response" are the ASN.1 module's PDUs.

The "Request" and "Response" PDUs are always to be sent wrapped in KRB-PRIV messages, except for the "Error-Response" PDU which MUST be sent as KRB-ERROR e-data (see section 4.4.1) when AP exchanges fail, otherwise it MUST be sent wrapped in a KRB-PRIV.

The PDUs are described in section 6.

6 ASN1 Module

```
DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- Unsafe: IMPORT AP-REQ, AP-REP, KRB-ERROR, KRB-PRIV, PrincipalName,
--                           Realm, KerberosString, FROM KerberosV5Spec2

-- From [clarifications]
PrincipalName       ::= SEQUENCE {
        name-type   [0] Int32,
        name-string [1] SEQUENCE OF UTF8String
}
Realm               ::= UTF8String
-- NOTE WELL: Principal and realm names MUST be constrained by the
--            specification of the version of Kerberos V used by the
--            client.
--
-- [Perhaps PrincipalName should be a SEQUENCE of an optional name type
-- and a UTF8String, for simplicity.]
```

```
-- From [clarifications]
Int32        ::= INTEGER (-2147483648..2147483647)
UInt32       ::= INTEGER (0..4294967295)
```

```
-- Based on EncryptionKey type from [clarifications]
Key         ::= SEQUENCE {
        enc-type        [0] Int32,      -- from Kerberos
        key             [1] OCTET STRING,
        ...
}

Etype       ::= Int32   -- as in [clarifications]
-- Perhaps we should use an extensible CHOICE of Int32?

Language-Tag        ::= UTF8String -- Constrained by [RFC3066]

-- Use LangTaggedText instead of UTF8String for *-text fields and remove
-- "language" field?
--
-- LangTaggedText should be used as e-text for KRB-ERROR, at least in
-- extensions, perhaps in [clarifications]
LangTaggedText      ::= SEQUENCE {
        language        [0] Language-Tag OPTIONAL,
        text            [1] UTF8String,
        ...
}

Request         ::= [APPLICATION 0] SEQUENCE {
        pvno-major      [0] INTEGER DEFAULT 2,
        pvno-minor      [1] INTEGER DEFAULT 0,
        languages       [2] SEQUENCE OF Language-Tag OPTIONAL,
        targ-name       [3] PrincipalName OPTIONAL,
        targ-realm      [4] Realm OPTIONAL,
                -- If targ-name/realm are missing then the request
                -- applies to the principal of the client
        operation       [5] Op-req,
        ...
}

Response        ::= [APPLICATION 1] SEQUENCE {
        pvno-major      [0] INTEGER DEFAULT 2,
        pvno-minor      [1] INTEGER DEFAULT 0,
        language        [2] Language-Tag DEFAULT "i-default",
        result          [3] Op-rep OPTIONAL,
        ...
}

Error-Response  ::= [APPLICATION 2] SEQUENCE {
        pvno-major      [0] INTEGER DEFAULT 2,
        pvno-minor      [1] INTEGER DEFAULT 0,
        language        [2] Language-Tag DEFAULT "i-default",
        error-code      [3] ProtocolErrorCode,
```

```
        help-text       [4] UTF8String OPTIONAL,
        op-error        [5] Op-error OPTIONAL,
        ...
}
```

```
Op-req          ::= CHOICE {
        null                    [0] Req-null,
        change-pw               [1] Req-change-pw,
        set-keys                [2] Req-set-keys,
        get-pw-policy           [3] Req-get-pw-policy,
        get-princ-aliases       [4] Req-get-princ-aliases,
        get-supported-etypes    [5] Req-get-supported-etypes,
        ...
}

Op-rep          ::= CHOICE {
        null                    [0] Rep-null,
        change-pw               [1] Rep-change-pw,
        set-keys                [2] Rep-set-keys,
        get-pw-policy           [3] Rep-get-pw-policy,
        get-princ-aliases       [4] Rep-get-princ-aliases,
        get-supported-etypes    [5] Rep-get-supported-etypes,
        ...
}

Op-error        ::= CHOICE {
        null                    [0] Err-null,
        change-pw               [1] Err-change-pw,
        set-keys                [2] Err-set-keys,
        get-pw-policy           [3] Err-get-pw-policy,
        get-princ-aliases       [4] Err-get-princ-aliases,
        get-supported-etypes    [5] Err-get-supported-etypes,
        ...
}

ProtocolErrorCode           ::= ENUM {
        -- Remember, ASN.1 enums are zero-based
        generic-error,
        unsupported-major-version,
        unsupported-minor-version,
        unsupported-operation,
        authorization-failed,
        initial-ticket-required,
        target-principal-unknown,
        ...
}

--
-- Requests and responses
--

-- NULL request, much like ONC RPC's NULL procedure
Req-null    ::= NULL
```

```
Rep-null    ::= NULL

Err-null    ::= NULL
```

```
-- Change password
Req-change-pw   ::= SEQUENCE {
        old-pw          [0] UTF8String,
        new-pw          [1] UTF8String OPTIONAL,
        etypes          [2] SEQUENCE (1..) OF Etype OPTIONAL,
        ...
}

Rep-change-pw   ::= SEQUENCE {
        info-text       [0] UTF8String OPTIONAL,
        new-pw          [1] UTF8String OPTIONAL,
                                -- generated by the server if present
                                -- (and requested by the client)
        etypes          [2] SEQUENCE (1..) OF Etype OPTIONAL,
        ...
}

Err-change-pw   ::= SEQUENCE {
        help-text               [0] UTF8String OPTIONAL,
        code                    [1] ENUM {
                generic,
                wont-generate-new-pw,
                old-pw-incorrect,
                new-pw-rejected-generic,
                pw-change-too-soon,
                ...
        },
        suggested-new-pw        [2] UTF8String OPTIONAL,
        ...
}

-- Change/Set keys

Req-set-keys    ::= SEQUENCE {
        etypes          [0] SEQUENCE (1..) OF Etype,
        entropy         [1] OCTET STRING OPTIONAL,
                                -- The client can provide entropy for
                                -- the server's use while generating
                                -- keys.
        ...
}

Rep-set-keys    ::= SEQUENCE {
        info-text       [0] UTF8String OPTIONAL,
        kvno            [1] UInt32,
        keys            [2] SEQUENCE (1..) OF Key,
                                -- The server always makes the keys.
        aliases         [3] SEQUENCE OF SEQUENCE {
```

```
                name         [0] PrincipalName,
                realm        [1] Realm OPTIONAL,
                ...
        } OPTIONAL,
```

```
        ...
}

Err-set-keys    ::= SEQUENCE {
        help-text       [0] UTF8String OPTIONAL, -- Reason for rejection
        enctypes        [1] SEQUENCE of Etype OPTIONAL, -- supported enctypes
        code            [2] ENUM {
                etype-no-support,
                ...
        }
}

-- Get password policy
Req-get-pw-policy   ::= NULL

Rep-get-pw-policy   ::= SEQUENCE {
        help-text       [0] UTF8String OPTIONAL,
        policy-name     [1] UTF8String OPTIONAL,
        description     [2] UTF8String OPTIONAL,
        ...
}

Err-get-pw-policy   ::= NULL

-- Get principal aliases
Req-get-princ-aliases   ::= NULL

Rep-get-princ-aliases   ::= SEQUENCE {
        help-text       [0] UTF8String OPTIONAL,
        aliases         [1] SEQUENCE OF SEQUENCE {
                name        [0] PrincipalName,
                realm       [1] Realm OPTIONAL,
                ...
        } OPTIONAL,
        ...
}

Err-get-princ-aliases   ::= NULL

-- Get list of enctypes supported by KDC for new keys
Req-get-supported-etypes    ::= NULL

Rep-get-supported-etypes    ::= SEQUENCE OF Etype

Err-get-supported-etypes    ::= NULL

END
```

7 Descriptions of each protocol requests and responses

This section describes the semantics of each operation request and response defined in the ASN.1 module in section 6.

Requests and responses consist of an outer structure ("Request," "Response" and "Error-Response") containing fields common to all requests/responses, and an inner structure for fields that are specific to each operation's requests/responses.

Specifically, the outer Request structure has a field for passing a client's spoken (read) languages to the server. It also has two optional fields for identifying the operation's target principal's name and realm (if not sent then the server MUST use the client principal name and realm from the AP exchange as the target).

The Response and Error PDU' outer structures include a field indicating the language that the server has chosen for localization of text intended to be displayed to users.

All three PDUs, "Request," "Response," and "Error-Response" include a protocol version number and the two responses include an optional field through which the server can indicate which language, from the client's list, the server can "speak."

7.1 Null Request

The null request is intended for use with TCP; its purpose is similar to RPC null procedures and is akin to a "ping" operation.

7.2 Change Password

The change password request has two fields: old-pw (old password - required) and new-pw (new password - optional). The server MUST validate the old password and MUST check the quality of the new password, if sent, according to the password policy associated with the client's principal before accepting the request. If the client does not specify a new password the server MUST either generate one and return it in the response or reject the request with wont-generate-new-pw as the Err-change-pw message's error code.

If the server rejects a client's proposed new password it SHOULD include a description of the password quality policy in effect for the target principal and/or an explanation of what was wrong with the proposed password in the help-text field of the Err-change-pw message. Additionally, servers MAY include a randomly generated, but preferably user-friendly password in the suggested-new-pw field of Err-change-pw messages when the client's proposed new password violates the target principal's password quality policy; of course, any such suggested new password MUST pass the target principal's password quality policy.

Clients MAY specify key enctypes to set with new passwords, but generally SHOULD NOT do so. If a client requests specific enctypes then the server MUST NOT create keys from the new password of any enctype other than those requested by the client.

Servers MAY indicate the enctypes of the keys created with new passwords, but SHOULD NOT do so unless the client requested specific enctypes - in which case the server MUST include the new key enctypes in the change password response.

7.3 Set Keys Requests

The set keys request consists of a sequence of key enctypes and an optional OCTET STRING of client-provided entropy.

The server generates keys of the requested enctypes and returns them. The server MAY utilize some, all or none of the client-provided entropy, if any, to generate the keys, but the server SHOULD input some entropy in the process.

The server SHOULD also include a list of the target principal's aliases, if there are any.

7.5 The Get Policy Request

It is common for sites to set policies with respect to password quality. It is beyond the scope of this document to describe such policies. However, it is reasonable for password policies to have names and as such for this protocol to associate named password quality policies with principals. It may also be reasonable for users to learn of their password quality policies.

The protocol therefore provides an operation for retrieving the name and/or description of the password policy that applies to the target principal name.

Management of password quality policies' actual content is beyond the scope of this protocol.

7.6 The Get Aliases Request

This request allows a client to obtain a list of aliases associated with a principal so that the client can properly configure the principal's "keytab."

Principal aliases are principal names for which the KDC will issue tickets (with the alias being either the client or target principal name of such tickets) using the same key as the "canonical" principal name, but without canonicalizing the aliased names in KDC exchanges.

7.7 The Get Supported Enctypes Request

This request allows a client to learn of the target principal's realm's supported enctypes.

8 IANA Considerations

...

9 Security Considerations

Implementors and site administrators should note that the redundancy of UTF-8 encodings varies by Unicode codepoint used. Password quality policies should, therefore, take this into account when estimating the amount of redundancy and entropy in a proposed new password. [?? It's late at night - I think this is correct.]

Kerberos set/change password/key protocol major version negotiation cannot be done securely. A downgrade attack is possible against clients that attempt to negotiate the protocol major version to use with a server. It is not clear at this time that the attacker would gain much from such a downgrade attack other than denial of service (DoS) by forcing the client to use a protocol version which does not support some feature needed by the client (Kerberos V in general is subject to a variety of DoS attacks anyways [RFC1510]).

[More text needed]

10 Description of TLV Encoding of Sample Subsets of the Protocol

This section provides example descriptions of the TLV DER encodings of some requests and responses. This section is not intended to be authoritative and implementors are encouraged to base their implementations on the ASN.1 syntax given in section 6. These TLV descriptions are given here in the interest of promoting implementation of this protocol even by implementors who do not have access to ASN.1 development tools.

Tags are described as T(<tag>) where <tag> is a letter denoting the tag type (u for UNIVERSAL, a for APPLICATION, c for CONTEXT and p for PRIVATE) and a number or universal type name.

Lengths and values are described as L{<value>}, where <value> is a description of the encoding of the value, except for scalar UNIVERSAL types, where <value> shall be '<' description of value '>'.

Optional fields are enclosed in square brackets ('[' and ']').

Repetition is denoted by ellipsis ("...").

Extensibility is denoted by "[...]".

Comments are introduced by "--" as in ASN.1

10.1 TLV encoding of the Null request and response

```
-- Null Request
-- Outer application tag
```

```
   T(a0)L{T(uSEQUENCE)L{
     -- "preamble"
     -- pvno-major == 2 so it is left out
```

```
     -- pvno-minor == 0 so it is left out
     -- optional languages list
     [T(c2)L{
       T(uSEQUENCE)L{
         T(uUTF8String)L{<language tag>}
         ...
       }
     }]
     -- optional targ-name
     [T(c3)L{
       Tc(uSEQUENCE)L{
         -- name-type
         T(c0)L{T(uINTEGER)L{<name-type>}}
         -- name-string
         T(c1)L{
           T(uSEQUENCE)L{
             [T(uUTF8String)L{<component name>}]
             ...
           }
         }
       }
     }]
     -- optional targ-realm
     [T(c4)L{T(uUTF8String)L{<realm name>}}]
     -- end of preamble

     -- operation choice tag
     T(c5)L{
       -- null CHOICE (this tag indicates the CHOICE taken; replace this
       -- TLV with the TLV for any operation to get the Request encoding
       -- of that operation)
       T(c0)L{
         -- Req-null (this is the encoding of the value of the CHOICE
         -- taken); NULL has no LV part.
         T(uNULL)
       }
     }
     -- extensions
     [...]
   }}

   -- Null Response
   -- Outer application tag
   T(a1)L{T(uSEQUENCE)L{
     -- "preamble"
     -- pvno-major == 2 so it is left out
     -- pvno-minor == 0 so it is left out
     -- optional language
     [T(c1)L{T(uUTF8String)L{<language tag>}}]
```

```
     -- end preamble
     -- operation choice tag
     T(c2)L{
       -- null CHOICE
```

```
        T(c0)L{
          T(uNULL)
        }
      }
    }}
```

10.2 TLV encoding of Error-Response

```
    -- Error Response
    -- Outer application tag
    T(a1)L{T(uSEQUENCE)L{
      -- "preamble"
      -- pvno-major == 2 so it is left out
      -- pvno-minor == 0 so it is left out
      -- optional language
      [T(c2)L{T(uUTF8String)L{<language tag>}}]
      -- end preamble
      -- error code
      T(c3)L{T(uENUM)L{<error code}}
      T(c4)L{T(uUTF8String)L{<error text}}
      -- optional CHOICE
      T(c5)L{
        -- CHOICE TLV goes here
        T(c<choice taken>)L{<value of CHOICE taken>}
      }
      -- extensions
      [...]
    }}
```

10.3 TLV encoding of the change password requests and responses

```
    -- Req-change-pw
      -- choice tag
      T(c1)L{
        T(uSEQUENCE)L{
          -- old password
          T(c0)L{T(uUTF8String)L{<password string>}}
          -- new password (optional; if missing server must generate it)
          [T(c1)L{T(uUTF8String)L{<password string>}}]
          -- extensions
          [...]
        }
      }

    -- Rep-change-pw
      -- choice tag
      T(c1)L{
        T(uSEQUENCE)L{
          -- optional informational text
```

```
[T(c0)L{T(uUTF8String)L{<text>}}]
-- new password (optional; see section 6)
[T(c1)L{T(uUTF8String)L{<new password>}}]
-- extensions
```

```
          [...]
        }
      }

    -- Err-change-pw
      -- choice tag
      T(c1)L{
        T(uSEQUENCE)L{
          -- optional help text
          [T(c0)L{T(uUTF8String)L{<text>}}]
          -- error code
          T(c1)L{T(uENUM)L{<error code>}}]
          -- extensions
          [...]
        }
      }
```

10.4 TLV encoding of Change Keys requests and responses

```
    -- Req-set-keys
      -- choice tag
      T(c1)L{
        T(uSEQUENCE)L{
          -- new key enctypes
          T(c0)L{T(uSEQUENCE)L{
            T(uINTEGER)L{<etype integer>},
            ...
          }}
          -- optional entropy
          [T(c1)L{T(uOCTET STRING)L{<entropy>}}]
          -- extensions
          [...]
        }
      }

    -- Rep-set-keys
      -- choice tag
      T(c1)L{
        T(uSEQUENCE)L{
          -- optional informational text
          [T(c0)L{T(uUTF8String)L{<text>}}]
          -- new kvno
          T(c1)L{T(uINTEGER)L{<new kvno>}}
          -- new keys
          T(c2)L{T(uSEQUENCE)L{
            -- first key
            T(uSEQUENCE)L{
              T(uINTEGER)L{<etype>}
              T(uOCTET STRING)L{<key>}
```

```
      -- extensions to Key
      [...]
    }
    -- additional keys, if any
```

```
          ...
        }}
        -- optional aliases
        [T(c3)L{T(uSEQUENCE)L{
          -- first alias
          T(uSEQUENCE)L{
            -- principal name
            T(uSEQUENCE)L{
              T(uUTF8String)L{<component1>},
              -- components 2..N, if any
              ...
            }
            T(uUTF8String)L{<realm name>}
            -- extensions
            [...]
          }
          -- additional aliases, if any
          ...
        }}]
        -- extensions
        [...]
      }
    }

 -- Err-set-keys
   -- choice tag
   T(c1)L{
     T(uSEQUENCE)L{
       -- optional help text
       [T(c0)L{T(uUTF8String)L{<text>}}]
       -- KDC supported enctypes
       [T(c1)L{T(uSEQUENCE)L{
         T(uINTEGER)L{<etype integer>},
         ...
       }}]
       -- error code
       T(c2)L{T(uENUM)L{<error code>}}]
       -- extensions
       [...]
     }
   }
```

11 Acknowledgements

The authors would like to thank Sam Hartman, Paul W. Nelson and Marcus Watts for their assistance.

The authors thank Ken Raeburn, Tom Yu, Martin Rex, Sam Hartman, Tony Andrea, Paul W. Nelson, Marcus Watts and other participants from the

IETF Kerberos Working Group for their input to the document.

12 References

12.1 Normative References

[RFC2026]
S. Bradner, RFC2026: "The Internet Standard Process - Revision 3," October 1996, Obsoletes - RFC 1602, Status: Best Current Practice.

[RFC2119]
S. Bradner, RFC2119 (BCP14): "Key words for use in RFCs to Indicate Requirement Levels," March 1997, Status: Best Current Practice.

[X680]
Abstract Syntax Notation One (ASN.1): Specification of Basic Notation, ITU-T Recommendation X.680 (1997) | ISO/IEC International Standard 8824-1:1998.
http://www.itu.int/ITU-T/studygroups/com17/languages/X680_0702.pdf

[X690]
ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), ITU-T Recommendation X.690 (1997)| ISO/IEC International Standard 8825-1:1998.
http://www.itu.int/ITU-T/studygroups/com17/languages/X690_0702.pdf

[RFC1510]
J. Kohl and B. C. Neuman, RFC1510: "The Kerberos Network Authentication Service (v5)," September 1993, Status: Proposed Standard.

[clarifications]
RFC-Editor: To be replaced by RFC number for draft-ietf-krb-wg-kerberos-clarifications.

[k5stringprep]
RFC-Editor: To be replaced by RFC number for draft-ietf-krb-wg-utf8-profile.

[RFC3066]
H. Alvestrand, RFC3066 (BCP47): "Tags for the Identification of Languages," January 2001, Obsoletes RFC1766, Status: Best Current Practice.

[KPASSWDv1]
(Reference needed!)

12.1 Informative References

[RFC3244]
M. Swift, J. Trostle, J. Brezak, RFC3244: "Microsoft Windows 2000

Kerberos Change Password and Set Password Protocols," February 2002, Status: Informational.

13 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

Nicolas Williams
Sun Microsystems
5300 Riata Trace Ct
Austin, TX 78727
Email: nicolas.williams@sun.com

14 Notes to the RFC Editor

This document has two KRB WG drafts as normative references and cannot progress until those drafts progress, but no other draft depends on this one.

developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.