NETWORK WORKING GROUP
Internet-Draft
Expires: May 7, 2009

N. Williams
Sun

November 3, 2008

Kerberos Set/Change Key/Password Protocol Version 2
draft-ietf-krb-wg-kerberos-set-passwd-08.txt

Status of this Memo

This Internet-Draft will expire on March 27, 2009.

Copyright Notice

Abstract

This document specifies an extensible protocol for setting keys and changing the passwords of Kerberos V principals.

Table of Contents

1. Introduction

Up to this point Kerberos V has lacked a single, standard protocol for changing passwords and keys. While several vendor-specific protocols exist for changing Kerberos passwords/keys, none are properly internationalized and all are incomplete in one respect or another and none are sufficiently extensible to cope with new features that may be added to Kerberos V at some future time.

This document defines a protocol that is somewhat backward-compatible with the "kpasswd" protocol 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. The Protocol

The structure of the protocol is quite similar to that of typical Remote Procedure Call (RPC) protocols. Each transaction consists of a data structure specific to an operation which is then wrapped in a data structure which is general to all operations of the protocol. These data structures are defined with the Abstract Syntax Notation 1 (ASN.1) [CCITT.X680.2002] and they are encoded using the Distinguished Encoding Rules (DER) [CCITT.X690.2002].

All protocol data is wrapped KRB-PRIV messages, or, in some cases, a KRB-ERROR, and framed in a header that is backwards compatible with [RFC3244].

3.1 Transports

The service supports only connection-oriented transports, specifically TCP, and SHOULD accept requests on TCP port 464, the same as in [RFC3244].

3.1.1 Protocol Framing

Requests and responses are exchanged using the same framing as in [RFC3244], but with the following differences:

- the protocol number field MUST be set to 0x2 (not 0xff80 or 0x1)
- the 'AP-REQ length' field of the request can be set to 0x0, in which case the 'AP-REQ' field of the request is excluded
- the 'KRB-PRIV' field of the request and reply is mutually exclusive with the 'AP-REQ' field of the request
- the 'AP-REP length' field of the reply can be set to 0x0, in which case the 'AP-REP' field of the reply is excluded
- all errors MUST be sent in a KRB-PRIV if the client's AP-REQ can be or has been accepted by the server
- any KRB-ERROR messages are framed and sent in the 'AP-REP' field of the reply

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 or MUST be a KRB-ERROR.

Subsequent messages exchanged on the same TCP connection 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 the ticket last used for authentication expires or when the server responds with an error indicating that the client must re-authenticate.

3.2 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, defined herein.

3.2.1 Protocol Major Version Negotiation

Version 2 clients that also support other versions, such as 0xff80, as in [RFC3244], SHOULD attempt to use version 2 of the protocol first.

Servers which do not support version 2 of this protocol typically include their preferred version number in the reply and/or may include a reply in the e-data of a KRB-ERROR, or in a KRB-PRIV with a status code of KRB5_KPASSWD_MALFORMED.

Note that some [RFC3244] server implementations close the TCP connection without returning any other response. Note also that there is no integrity protection for the major version number in the protocol framing or for any data in a KRB-ERROR.

As a result change password protocol major version negotiation is subject to downgrade attacks. Therefore major version negotiation is NOT RECOMMENDED.

Where the server indicates that it does not support version 2, the client MAY, but SHOULD NOT, unless configured to do so, fall back on another major version of this protocol.

Version 2 servers MAY 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 3.1.1, using an AP-REP and KRB-PRIV if the client's AP-REQ can be accepted, or a KRB-ERROR otherwise and using a ProtocolErrorCode value of unsupported-major-version.

It is expected that implementations of as yet unspecified future major versions of this protocol will be required to support version 2 integrity protected error replies for properly indicating no support for version 2 of the protocl. We also hope that no further major versions of this protocol will be needed.

3.2.2 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.

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 TCP session.

See Section 3.6 for further description of the protocol's extensibility and its relation to protocol minor versions and the negotiation thereof.

3.3 Use of Kerberos V and Key Exchange

This protocol makes use of messages defined in [RFC4120]. Specifically, AP-REQ, AP-REP, KRB-ERROR and KRB-PRIV.

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

Clients SHOULD use "kadmin/setpw" as the principal name of the server for all requests except when changing the client principal's own expired password, for which they should use "kadmin/changepw". The "kadmin/changepw" service exists to allow KDCs to limit principals with expired passwords to getting initial tickets to the password changing service only and only for changing expired passwords.

Servers MUST limit clients that used the "kadmin/changepw" service principal name to changing the password of the client principal.

The client MUST request mutual authentication and the client MUST MUST request the use of sequence numbers.

Servers MAY force the use of INITIAL or fresh tickets for any requests -- see Section 4.3. Traditionally users with expired passwords are allowed only INITIAL tickets for the password changing server, therefore clients MUST support the use of INITIAL tickets.

Servers MUST specify a sub-session key.

The encrypted part of KRB-PRIVs MUST be encrypted with the server's sub-session key and key usage 20 (client->server) or 21 (server->client).

After each new AP exchange the client and server MUST destroy the session keys, if any, resulting from the previous AP exchange.

3.4 Use of ASN.1

This protocol's messages are defined in ASN.1, using only features from [CCITT.X680.2002]. All ASN.1 types defined herein are to be encoded in DER [CCITT.X690.2002]. A complete ASN.1 module is given in Section 5.

The DER encoding of the ASN.1 PDUs are exchanged wrapped in a KRB-PRIV as described above and/or as e-data in KRB-ERROR messages.

3.5 Internationalization

This protocol's request PDU carries an optional field indicating the languages spoken by the client user; the client SHOULD send its list of spoken languages to the server (once per-TCP session).

The server SHOULD localize all strings intended for display to users to a language in common with the languages spoken by the client user.

Strings for Kerberos principal and realm names used in this protocol are be constrained as per [RFC4120].

3.5.1 Normalization Forms for UTF-8 Strings

Because Kerberos V [RFC4120] restricts principal names, realm names and passwords to IA5String, this protocol uses UTF8String with an extensible constraint to IA5String.

Future versions of Kerberos may relax this constraint; if so then a minor version of this protocol should relax this constraint accordingly.

3.5.2 Language Negotiation

The server MUST pick a language from the client's input list or the default language tag (see [RFC4646] and [RFC4647]) 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.

Servers MUST cache the optional language tag lists from prior requests for use with subsequent requests that exclude the language tag list. Clients MAY expect such server behaviour and send the language tag lists only once per-TCP session. Clients SHOULD send the server the language tag list at least once.

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

3.6 Protocol Extensibility

The protocol is defined in ASN.1 and uses extensibility markers in some places, particularly extensible CHOICEs. As such, the module presented herein can be extended within the framework of [CCITT.X680.2002].

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 3.2.2 for the minor version negotiation.

Servers SHOULD ignore unknown additions to the ASN.1 types, in initial requests, where the syntax allows them, except for extensions to the "Op-req" type, which MUST result in an error.

Servers MUST respond with an error (ProtocolErrorCode value of proto-unsupported-operation) to clients that use operations unknown to the server.

3.7 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. Client and server implementations MAY implement subsets of the overall protocol by removing some alternatives to the Op-req, Op-rep and Op-err CHOICEs from the ASN.1 module given in Section 5.

For example, it should be possible to have a password-change only client that cannot set principal's keys - and vice versa.

The following operations are REQUIRED to implement:

- Null
- Change Kerberos Password (with commit == TRUE)
- Set Kerberos Password (with commit == TRUE)
- Generate Kerberos Keys
- Get Password Quality Policy

4. Protocol Elements

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

- change password (or enctypes and string-to-key parameters)
- set password (administrative)
- set new keys
- generate new keys
- get new, un-committed keys
- commit new keys
- get password policy name and/or description of principal
- list aliases of a principal
- list enctypes and version of Kerberos V supported by realm

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 key databases 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 other means, such as through directories or other Kerberos administration protocols.

The individual operations are described in Section 4.3.

4.1 Password Quality Policies

Servers may reject new user passwords for failing to meet password quality policies. When this happens the server must be able to communicate the policy to the user so that the user may select a better password.

The protocol allows for two ways to do this:

- through error codes that identify specific password quality policies known;
- through localized error strings.

The use of localized error strings allows servers to convey non-standard password quality policies to users, but it requires server-side message catalogs for localization and support for language negotiation. The use of error codes only allows for standard password quality policies that clients must be aware of a priori, therefore use of error codes requires the distribution of new message catalogs to clients whenever new error codes are added; this simplifies servers at the expense of password quality extensibility.

When a server would reject a password due to its failing to meet a standard password quality policy the the server MUST use the appropriate error codes (see below) and the server SHOULD send a localized error message to the user.

When a server would reject a password due to its failing to meet a non-standard password quality policy (one not described herein) the the server MUST send a localized error message to the user.

4.1.1 Standard Password Quality Policies

- pwq-too-soon

  It is too soon for the principal to change its password or long-term keys.

- pwq-history

  The new password is one that the principal has used before or is too similar to a password that it has used before.

- pwq-too-short

  The new password is too short.

- pwq-dictionary-words

  The new password is or contains words that are in the dictionary.

- pwq-prohibited-codepoints

  The new password contains prohibited codepoints.

- pwq-need-more-char-classes

  The new password does not have characters from enough character classes (lower-case letters, upper-case letters, digits, punctuation, etc...).

4.2 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 3.3) when AP exchanges fail, otherwise it MUST be sent wrapped in a KRB-PRIV.

The ASN.1 syntax for the PDUs is given in Section 5.

Note that the first field of each PDU is the major version of the protocol, defaulted to 2, meaning that it is never included in version 2 exchanges. Similarly, the second field of each PDU is the minor version, defaulted to 0.

The request, responses and error PDUs consist of an outer structure ("Request," "Response" and "Error-Response") containing fields common to all requests, responses and errors, respectively, and an inner structure for fields that are specific to each operation's requests/responses. The inner structure is optional in the case of the Error-Response PDU and need not be included when generic errors occur for which there is a suitable ProtocolErrorCode.

Specifically, the outer Request structure has a field for passing a client user's spoken (read) languages to the server. It also has two optional fields for identifying the requested operation's target principal's name and realm (if not sent then the server MUST use the client's principal name and realm as the target). A boolean field for indicating whether or not the request should be dry-run is also included; dry-runs can be used to test server policies, and servers MUST NOT modify any principals when processing dry-run requests.

The Response and Error PDUs' outer structures include a field indicating the language that the server has chosen for localization of text intended to be displayed to users; this field is defaulted to "i-default". This language tag applies to all UTF8 strings in the inner structure (Op-rep and Op-err) that are meant to be displayed to users.

The protocol error codes are:

- proto-generic-error

  An operation-specific error ocurred, see the inner Op-error.

- proto-format-error

  The server failed to parse a message sent by the client.

- proto-unsupported-major-version

  The server does not support the major version of this protocol requested by the client.

- proto-unsupported-minor-version

  The server does not support the minor version of this protocol requested by the client.

- proto-unsupported-operation

  The server does not support the operation requested by the client.

- proto-wrong-service-principal

  Use kadmin/setpw for the server's principal name.

- proto-re-authentication-required

  The server demands that the client re-authenticate through a new AP exchange.

- proto-initial-ticket-required

  Use of an INITIAL ticket is required for the requested operation.

- proto-client-and-target-realm-mismatch

  The server requires that the client's principal name and the target principal of the operation share the same realm name.

- proto-target-principal-unknown

  The target of the client's operation is not a valid principal.

- proto-authorization-failed

The client principal is not authorized to perform the requested operation.

- proto-fresh-ticket-required

  The server would like the client to re-authenticate using a fresh ticket.

4.3 Operations

This section describes the semantics of each operation request and response defined in the ASN.1 module in Section 5.

4.3.1 Null

NAME

null - Null or "ping" operation

DESCRIPTION

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

ERRORS

None.

4.3.2 Change Kerberos Password

NAME

change-pw - Change password operation

SYNOPSIS

```
Req-change-pw(old-pw, [languages], [new-pw],
              commit?, [etypes]) ->
   Rep-change-pw([info-text], [new-pw], [etypes]) |
   Err-change-pw([help-text], error code, [error info])
```

DESCRIPTION

Change a principal's password.

The change password request has one required, three optional and one defaulted arguments: "old-pw" (required), "languages," "new-pw", "commit" (defaults to "TRUE") and "etypes", corresponding to the target principal's old password, its preferred languages, its new password, a boolean indicating whether or not to make the new long-term key available for immediate use, and the desired enctypes for the new long-term keys.

The server MUST validate the old password and MUST check the quality of the new password, if sent, according the password quality policy associated with the target principal.

If the old and new passwords in the request are the same strings, and the principal is not currently required to change its password, then the server MAY permit the password change as way to change a principal's enctypes and string-to-key parameters. This feature provides a way to, for example, add enctypes to a principals' password-derived long-term keys without forcing a password change following an upgrade to the KDC that adds support for new enctypes.

A client MAY request that the server generate a new password by excluding the new password from its request, in which case the server MUST either generate a new password or respond with an error indicating that it does not support this feature.

Server-generated passwords MUST meet the target principal's password quality policy. It is RECOMMENDED that server-generated passwords be user-friendly, that is, memorable and that the target principal's preferred languages be taken into account by the password generation alogrithm used by the server.

When "commit" is true the KDC makes the keys derived from the new password available for issueing tickets encrypted in them immediately. Otherwise the client MUST follow up with a commit-keys request to make the keys available. This feature is useful for changing keys shared by multiple hosts, in clustered services, for example, in an atomic manner.

RETURN

Upon successful password changes the server responds with a Rep-change-pw. The fields of Rep-change-pw are all optional and

include:

- 'info-text' which the server can use to send a message to the user such as "Your new password will expire in 90 days," for example.

- 'new-pw' which the server MUST include if the client requested that the server generate a new password; generated passwords MUST pass the target principal's password quality policy.

- 'etypes' which the server MAY include to indicate which types of long-term keys it created for the target principal and which the server MUST include if the client specified a set of enctypes in its request.

ERRORS

The server may respond to change password requests with protocol or operation errors.

All operation errors include an optional 'help-text' field by which the server can describe the error in a human-readable, localizaed string.

Change password error codes include:

- generic-error

- old-pw-incorrect

- wont-generate-new-pw

  The server will not generate a new password for this principal or does not support password generation in general.

- new-pw-rejected-generic

  The client's proposed new password failed the target principal's password quality policy.

  The server MUST include a description of the password quality policy or aspect of it that the client's proposed new password failed to meet.

  The server MAY generate and send a new password that the client can then use as a new password and which is guaranteed to pass the target principal's current password quality

policy.

The server MAY include a set of policy error code hints.

- etype-not-supported

The client requested an enctype that the KDC does not support.

4.3.3 Set Kerberos Password

NAME

set-pw - Set password operation

SYNOPSIS

```
Req-set-pw([languages], [new-pw], commit?, [etypes]) ->
   Rep-set-pw([info-text], [new-pw], [etypes]) |
   Err-set-pw([help-text], error code, [error info])
```

DESCRIPTION

Administratively set a principal's password.

The set password request has three optional and one defaulted arguments: "languages", "new-pw," "commit" (defaulted to "TRUE") and "etypes", corresponding to the target principal's preferred languages, new password, a boolean indicating whether or not to make the new long-term key available for immediate use, and the desired enctypes for the new long-term keys.

The server MUST check the quality of the new password, if sent, according the password quality policy associated with the target principal.

The server SHOULD require that the client use the change-pw operation instead of set-pw when the client principal and the target principal are the same.

A client MAY request that the server generate a new password by excluding the new password from its request, in which case the server MUST either generate a new password or respond with an

error indicating that it does not support this feature.

Server-generated passwords MUST meet the target principal's password quality policy. It is RECOMMENDED that server-generated passwords be user-friendly, that is, memorable and that the target principal's preferred languages be taken into account by the password generation alogrithm used by the server.

When "commit" is true the KDC makes the keys derived from the new password available for issueing tickets encrypted in them immediately. Otherwise the client MUST follow up with a commit-keys request to make the keys available. This feature is useful for changing keys shared by multiple hosts, in clustered services, for example, in an atomic manner.

RETURN

Upon successfully setting a password the server responds with a Rep-set-pw. The fields of Rep-set-pw are all optional and include:

- 'info-text' which the server can use to send a message to the user such as "Your new password will expire in 90 days," for example.

- 'new-pw' which the server MUST include if the client requested that the server generate a new password; generated passwords MUST pass the target principal's password quality policy.

- 'etypes' which the server MAY include to indicate which types of long-term keys it created for the target principal and which the server MUST include if the client specified a set of enctypes in its request.

ERRORS

The server may respond to set password requests with protocol or operation errors.

All operation errors include an optional 'help-text' field by which the server can describe the error in a human-readable, localizaed string.

Set password error codes include:

- generic-error

- use-change-pw

  The server demands that the client use the change-pw operation for the target principal of the set-pw request.

- wont-generate-new-pw

  The server will not generate a new password for this principal or does not support password generation in general.

- new-pw-rejected-generic

  The client's proposed new password failed the target principal's password quality policy.

  The server MUST include a description of the password quality policy or aspect of it that the client's proposed new password failed to meet.

  The server MAY generate and send a new password that the client can then use as a new password and which is guaranteed to pass the target principal's current password quality policy.

  The server MAY include a set of policy error code hints.

- etype-not-supported

  The client requested an enctype that the KDC does not support.

4.3.4 Set Kerberos Keys

NAME

set-keys - Set a principal's long-term keys

SYNOPSIS

```
Req-set-keys(new-keys, commit?, [isupport]) ->
   Rep-set-keys([info-text], kvno, aliases, [isupport])
```

DESCRIPTION

The set-keys request consists of two required fields and one optional field: "new-keys", "commit" (a boolean field - see below) and "isupport", an optional field for indicating to the KDC what Kerberos V features are supported by the target principal.

When "commit" is true the KDC makes the new keys available for issueing tickets encrypted in them immediately. Otherwise the client MUST follow up with a commit-keys request to make the keys available. This feature is useful for changing keys shared by multiple hosts, in clustered services, for example, in an atomic manner.

If a principal has keys are awaiting commitment when a new set-keys request for that principal s made then the KDC MUST overwrite the deferred keys.

RETURN

For successful set-keys operations the server returns:

- Informational text, optional.
- The new kvno for the target principal.
- A list of aliases of the target principal known to the KDC (optional).
- The set of Kerberos V features supported by the KDC (optional).

ERRORS

The server may respond with the following errors:

- generic
- deferred-commit-no-support
- etype-no-support

4.3.5 Generate Kerberos Keys

NAME

gen-keys - Generate long-term keys for a principal

SYNOPSIS

```
Req-gen-keys(etypes, [entropy], commit?, [isupport]) ->
   Rep-set-keys([info-text], key, kvno, aliases, [isupport])
```

DESCRIPTION

The gen-keys is similar to the set-keys request but differs in that the server generates keys of client-requested enctypes, rather than the client providing specific keys.

The gen-keys request consists of two required fields and two optional fields: "etypes" (the enctypes of the new keys), "entropy", "commit" and "isupport".

If a principal has keys are awaiting commitment when a new set-keys request for that principal s made then the KDC MUST overwrite the deferred keys.

RETURN

For successful set-keys operations the server returns:

- Informational text, optional.
- The new kvno for the target principal.
- The new key (only one is needed).
- A list of aliases of the target principal known to the KDC (optional).
- The set of Kerberos V features supported by the KDC (optional).

ERRORS

The server may respond with the following errors:

- generic

- deferred-commit-no-support
- etype-no-support

4.3.6 Get New Keys

NAME

get-keys - Retrieve a principal's uncommitted keys

SYNOPSIS

```
Req-get-keys(kvno) ->
   Rep-get-keys([info-text], keys, aliases, [isupport]) |
   Err-get-keys([help-text], error code, [error info])
```

DESCRIPTION

This request allows a client to get the keys set or generated in a previous set-keys or gen-keys request with deferred commitment.

RETURN

If the target principal and kvno correspond to uncommitted keys the server SHOULD respond with the actual keys that would be set by a subsequent commit-keys request. Otherwise the server MUST respond with an error (meaning that this operation cannot be used to extract keys from the KDC that may be in use).

ERRORS

- generic
- kvno-committed
- no-such-kvno

4.3.7 Commit New Keys

NAME

commit-keys - Commit a principal's new long-term keys

SYNOPSIS

```
Req-commit-keys(kvno) ->
   Rep-commit-keys() |
   Err-commit-keys([help-text], error code, [error info])
```

DESCRIPTION

The commit-keys operation allows a client to bring a principal's new keys into use at the KDC.

Clients should make a commit-keys request corresponding to a deferred commitment set-keys/gen-keys operation as soon as the local key database for the target principal is updated.

The target principal name and the kvno MUST match those from a prior set-keys or gen-keys operation.

Servers MAY expire delayed key commitments at will. Servers SHOULD expire uncommitted new keys after a reasonable amount of time (600 seconds is RECOMMENDED).

Servers MUST respond to new set-keys requests for principals with pending, uncommitted new keys by expiring the uncommitted new keys and proceeding as if there had been no expired new keys.

ERRORS

- generic
- op-kvno-expired
- op-kvno-unknown
- new-keys-conflict (A set-keys or gen-keys request succeeded subsequent to the request that matches this {principal, kvno} tuple.)

4.3.8 Get Password Quality Policy

NAME

get-pw-policy - Retrieve a principal's password policy

SYNOPSIS

```
Req-get-pw-policy() ->
   Rep-get-pw-policy([policy name], [policy description])
```

DESCRIPTION

Returns a description of the target principal's associated password quality policy, if any, as a list of localized UTF8String values.

Clients can use this operation in conjunction with the change-pw operation to obtain text that can be displayed to the user before the user actually enters a new password.

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

ERRORS

No operation errors are defined.

4.3.9 Get Principal Aliases

NAME

get-princ-aliases - Retrieve a principal's aliases

SYNOPSIS

```
Req-get-princ-aliases() ->
   Rep-get-princ-aliases(aliases)
```

DESCRIPTION

Returns a list of aliases of the target principal.

ERRORS

No operation-specific errors.

4.3.10 Get Realm's Supported Kerberos V Version and Features

NAME

get-realm-krb5-support - List features supported by the realm

SYNOPSIS

```
Req-get-realm-krb5-support() ->
   Rep-get-realm-krb5-support(isupport)
```

DESCRIPTION

Returns set of Kerberos V features support by the target principal's realm's KDCs.

ERRORS

No operation-specific errors.

4.3.11 Retrieve Principal's S2K Params and Preferred Params

NAME

get-s2kparams

SYNOPSIS

```
Req-get-s2kparams() ->
   Rep-get-s2kparams(princ-s2kparams)
```

DESCRIPTION

Returns the string2key parameters for the principal's current password-derived long-term keys, if any (if there are none then the principal does not have a password-derived long-term key).

ERRORS

No operation-specific errors.

4.4 Principal Aliases

Applications that use Kerberos often have to derive acceptor principal names from hostnames entered by users. Such hostnames may be aliases, they may be fully qualified, partially qualified or not qualified at all. Some implementations have resorted to deriving principal names from such hostnames by utilizing the names services to canonicalize the hostname first; such practices are not secure unless the name service are secure, which often aren't.

One method for securely deriving principal names from hostnames is to alias principals at the KDC such that the KDC will issue tickets for principal names which are aliases of others. It is helpful for principals to know what are their aliases as known by the KDCs.

Note that changing a principal's aliases is out of scope for this protocol.

4.5 Kerberos V Feature Negotiation

Principals and realms' KDCs may need to know about additional Kerberos V features and extensions that they each support. Several

operations (see above) provide a way for clients and servers to exchange such infomration, in the form of lists of types supported for the various typed holes used in Kerberos V.

5. ASN.1 Module

```
DEFINITIONS EXPLICIT TAGS ::= BEGIN
--
-- Note:  EXPLICIT tagging is in use by default throughout this
--        module.

-- From [clarifications] with modifications
PrincipalName           ::= SEQUENCE {
     name-string [1] SEQUENCE OF UTF8String (IA5String, ...)
}
Realm                   ::= UTF8String (IA5String, ...)
Salt                    ::= UTF8String (IA5String, ...)
Password                ::= UTF8String (IA5String, ...)

-- 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 [clarifications]
Etype           ::= Int32
Etype-Info-Entry ::= SEQUENCE {
     etype          [0] Etype,
     salt           [1] Salt OPTIONAL,
     s2kparams      [2] OCTET STRING OPTIONAL
}
Key             ::= SEQUENCE {
     enc-type       [0] Etype,        -- from Kerberos
     key            [1] OCTET STRING
}

Language-Tag    ::= UTF8String -- Constrained by [RFC4646]

-- Empty, extensible SEQUENCEs are legal ASN.1
Extensible-NULL  ::= SEQUENCE {
     ...
}
```

```
   -- Kerberos clients negotiate some parameters relating to their peers
   -- indirectly through the KDC.  Today this is true of ticket session
   -- key enctypes, but in the future this indirect negotiation may also
   -- occur with respect to the minor version of Kerberos V to be used
   -- between clients and servers.  Additionally, KDCs may need to know
   -- what authorization-data types are supported by service principals,
   -- both, for compatibility with legacy software and for optimization.
   --
   -- Thesefore it is important for KDCs to know what features of
   -- Kerberos V each service principal supports.
   --
   -- In version 2.0 of this protocol the clients and servers may notify
   -- each other of their support for:
   --
   --  - enctypes
   --  - authorization data types
   --  - transited encoding data types
   --
   -- All authorization-data types defined in [clarifications] are
   -- assumed to be supported if the minor version is 1 and do not need
   -- to be included in the ad-type list.
   --
   -- Int32 is used for enctype and transited encoding data type
   -- identifiers.

   --
   -- An extensible CHOICE of Int32 is used for authorization data
   -- types.

   KerberosV-TR-ID             ::= Int32

   KerberosV-AD-ID             ::= CHOICE {
        ad-int                [0] Int32,
        ...
   }

   KerberosVSupportNego        ::= SEQUENCE {
        enc-types       [0] SEQUENCE OF Etype,
        ad-types        [1] SEQUENCE OF KerberosV-AD-ID OPTIONAL,
                                    -- authorization data types
        tr-enc-types    [2] SEQUENCE OF KerberosV-TR-ID OPTIONAL
                                    -- transited encoding types
   }

   Request                     ::= [APPLICATION 0] SEQUENCE {
        pvno-minor      [0] INTEGER DEFAULT 0,
        languages       [1] SEQUENCE OF Language-Tag OPTIONAL,
                -- Should be defaulted to the SEQUENCE of "i-default"
```

```
        targ-name       [2] PrincipalName OPTIONAL,
        targ-realm      [3] Realm OPTIONAL,
                -- If targ-name/realm are missing then the request
                -- applies to the principal of the client
        dry-run         [4] BOOLEAN DEFAULT FALSE,
        operation       [5] Op-req,
        ...
    }

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

    Error-Response              ::= [APPLICATION 2] SEQUENCE {
        pvno-minor      [0] INTEGER DEFAULT 0,
        language        [1] Language-Tag DEFAULT "i-default",
        error-code      [2] ProtocolErrorCode,
        help-text       [3] UTF8String OPTIONAL,
        op-error        [4] Op-err OPTIONAL,
        ...
    }

    Op-req                      ::= CHOICE {
        null                      [0] Req-null,
        change-pw                 [1] Req-change-pw,
        set-pw                    [2] Req-set-pw,
        set-keys                  [3] Req-set-keys,
        gen-keys                  [4] Req-gen-keys,
        get-keys                  [5] Req-get-keys,
        commit-keys               [6] Req-commit-keys,
        get-pw-policy             [7] Req-get-pw-policy,
        get-princ-aliases         [8] Req-get-princ-aliases,
        get-realm-krb5-support    [9] Req-get-realm-krb5-support,
        get-s2kparams             [10] Req-get-s2kparams,
        ...
    }

    Op-rep                     ::= CHOICE {
        null                     [0] Rep-null,
        change-pw                [1] Rep-change-pw,
        set-pw                   [2] Rep-set-pw,
        set-keys                 [3] Rep-set-keys,
        gen-keys                 [4] Req-gen-keys,
        get-keys                 [5] Req-get-keys,
        commit-keys              [6] Rep-commit-keys,
```

```
        get-pw-policy           [7] Rep-get-pw-policy,
        get-princ-aliases       [8] Rep-get-princ-aliases,
        get-realm-krb5-support  [9] Rep-get-realm-krb5-support,
        get-s2kparams           [10] Rep-get-s2kparams,
        ...
    }

    Op-err        ::= CHOICE {
        null                    [0] Err-null,
        change-pw               [1] Err-change-pw,
        set-pw                  [2] Err-set-pw,
        set-keys                [3] Err-set-keys,
        gen-keys                [4] Err-gen-keys,
        get-keys                [5] Err-get-keys,
        commit-keys             [6] Err-commit-keys,
        get-pw-policy           [7] Err-get-pw-policy,
        get-princ-aliases       [8] Err-get-princ-aliases,
        get-realm-krb5-support  [9] Err-get-realm-krb5-support,
        get-s2kparams           [10] Err-get-s2kparams,
        ...
    }

    ProtocolErrorCode           ::= ENUM {
        proto-format-error,
        proto-unsupported-major-version,
        proto-unsupported-minor-version,
        proto-unsupported-operation,      -- Request CHOICE tag unknown
        proto-generic-see-op-error,       -- See Op-error
        proto-wrong-service-principal,    -- Use kadmin/setpw
        proto-re-authentication-required,
        proto-initial-ticket-required,
        proto-client-and-target-realm-mismatch,
        proto-target-principal-unknown,
        proto-authorization-failed,
        proto-dry-run-not-permitted,
        proto-fresh-ticket-required,
        ...
    }

    -- These codes are hints for clients, primarily for when they are
    -- used for changing the passwords of automated principals; error
    -- replies carry password quality policy help text that is more
    -- appropriate for clients to display to users.
    PW-Quality-Codes        ::= ENUM {
        pwq-generic,
        pwq-too-soon,
        pwq-history,
        pwq-too-short,
```

```
        pwq-dictionary-words,
        pwq-prohibited-codepoints,
        pwq-need-more-char-classes,
        ...
    }

    --
    -- Requests and responses
    --

    -- NULL request, much like ONC RPC's NULL procedure - NOT extensible
    Req-null    ::= NULL

    Rep-null    ::= NULL

    Err-null    ::= NULL

    -- Change password
    Req-change-pw        ::= SEQUENCE {
        old-pw             [0] Password,
        languages          [1] SEQUENCE OF Language-Tag OPTIONAL,
        new-pw             [2] Password OPTIONAL,
        commit             [3] BOOLEAN DEFAULT TRUE,
        etypes             [4] SEQUENCE (1..MAX) OF Etype OPTIONAL
    }

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

    Err-change-pw        ::= SEQUENCE {
        help-text          [0] UTF8String OPTIONAL,
        error              [1] CHOICE {
            op-generic-error           [0] Extensible-NULL,
            op-old-pw-incorrect        [1] Extensible-NULL,
            op-wont-generate-new-pw    [2] Extensible-NULL,
            op-new-pw-rejected-generic [3] SEQUENCE {
                    policy                  [1] SEQUENCE OF UTF8String,
                    suggested-pw            [2] Password OPTIONAL,
                    policy-codes            [3] SET OF PW-Quality-Codes
                                                OPTIONAL
            }
            op-etype-not-supported     [4] SEQUENCE {
                    supported-etypes   [1] SEQUENCE OF Etype
```

```
          }
     }
}

-- Set password
Req-set-pw        ::= SEQUENCE {
     languages             [0] SEQUENCE OF Language-Tag OPTIONAL,
     new-pw                [1] Password OPTIONAL,
     commit                [2] BOOLEAN DEFAULT TRUE,
     etypes                [3] SEQUENCE (1..MAX) OF Etype OPTIONAL
}

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

Err-set-pw        ::= Err-change-pw

-- Set keys
Req-set-keys      ::= SEQUENCE {
     keys             [0] SEQUENCE OF Key,
     commit           [1] BOOLEAN DEFAULT TRUE,
                           -- TRUE  -> install keys now
                           --
                           -- FALSE -> require set-keys-commit
                           --          before issuing tickets
                           --          encrypted with these keys.
                           --
                           -- See commit-keys op
     isupport         [2] KerberosVSupportNego OPTIONAL
                           -- For future Kerberos V extensions KDCs
                           -- may need to know what krb5 version is
                           -- supported by individual service
                           -- principals.  This field provides a
                           -- way to tell the KDC what version of
                           -- Kerberos V the target principal
                           -- supports.
}

Rep-set-keys        ::= SEQUENCE {
     info-text          [0] UTF8String OPTIONAL,
     kvno               [1] UInt32,
     aliases            [2] SEQUENCE OF SEQUENCE {
             name       [0] PrincipalName,
```

```
                realm    [1] Realm OPTIONAL
        },
        isupport         [3] KerberosVSupportNego OPTIONAL
        -- Should there be ETYPE-INFO2 stuff here?
    }

    Err-set-keys        ::= SEQUENCE {
        help-text     [0] UTF8String OPTIONAL, -- Reason for rejection
        error         [1] CHOICE {
                op-generic                    [0] Extensible-NULL,
                op-deferred-commit-no-support [1] Extensible-NULL,
                op-etype-no-support           [2] SEQUENCE OF {
                        supported-etypes      [1] SEQUENCE OF Etype
                }
        }
    }

    -- Generate keys
    Req-gen-keys        ::= SEQUENCE {
        etypes           [0] SEQUENCE (1..MAX) OF Etype,
        entropy          [1] OCTET STRING OPTIONAL,
        commit           [2] BOOLEAN DEFAULT TRUE,
                              -- TRUE  -> install keys now
                              --
                              -- FALSE -> require set-keys-commit
                              --          before issuing tickets
                              --          encrypted with these keys.
                              --
                              -- See commit-keys op
        isupport         [3] KerberosVSupportNego OPTIONAL
                              -- For future Kerberos V extensions KDCs
                              -- may need to know what krb5 version is
                              -- supported by individual service
                              -- principals.  This field provides a
                              -- way to tell the KDC what version of
                              -- Kerberos V the target principal
                              -- supports.
    }

    Rep-gen-keys        ::= SEQUENCE {
        info-text        [0] UTF8String OPTIONAL,
        kvno             [1] UInt32,
        key              [2] Key,
        aliases          [3] SEQUENCE OF SEQUENCE {
                name          [0] PrincipalName,
                realm         [1] Realm OPTIONAL
        },
        isupport         [4] KerberosVSupportNego OPTIONAL
```

```
        -- Should there be ETYPE-INFO2 stuff here?
    }

    Err-gen-keys        ::= Err-set-keys

    -- Get un-committed key request
    Req-get-keys        ::= SEQUENCE {
        kvno              [0] UInt32
    }

    Rep-get-keys        ::= SEQUENCE {
        info-text         [0] UTF8String OPTIONAL,
        keys              [1] SEQUENCE OF Key,
        aliases           [2] SEQUENCE OF SEQUENCE {
                name           [0] PrincipalName,
                realm          [1] Realm OPTIONAL
        },
        isupport          [3] KerberosVSupportNego OPTIONAL
        -- Should there be ETYPE-INFO2 stuff here?
    }

    Err-get-keys      ::= SEQUENCE {
        help-text       [0] UTF8String OPTIONAL, -- Reason for rejection
        error           [1] CHOICE {
                op-generic         [0] Extensible-NULL,
                op-kvno-committed  [1] Extensible-NULL,
                op-no-such-kvno    [1] Extensible-NULL
        }
    }

    -- Commit a set keys request
    Req-commit-keys ::= SEQUENCE {
        kvno          [0] UInt32
    }

    Rep-commit-keys ::= Extensible-NULL

    Err-commit-keys ::= SEQUENCE {
        help-text     [0] UTF8String OPTIONAL, -- Reason for rejection
        error         [1] CHOICE {
                op-generic           [0] Extensible-NULL,
                op-kvno-expired      [1] Extensible-NULL,
                    -- The client took too long to respond.
                op-kvno-unknown      [2] Extensible-NULL,
                    -- The targ princ/kvno is invalid or unknown to the
                    -- server (perhaps it lost track of state)
                op-new-keys-conflict [3] Extensible-NULL
                    -- A new-keys/commit-keys request subsequent to the
```

```
                  -- new-keys that produced the kvno has completed
                  -- and incremented the principal's kvno
        }
   }

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

   Rep-get-pw-policy   ::= SEQUENCE {
        policy-name      [0] UTF8String OPTIONAL,
        description      [1] SEQUENCE OF UTF8String OPTIONAL
   }

   Err-get-pw-policy   ::= Extensible-NULL

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

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

   Err-get-princ-aliases        ::= Extensible-NULL

   -- Get list of enctypes supported by KDC for new keys
   Req-get-realm-krb5-support   ::= Extensible-NULL

   Rep-get-realm-krb5-support   ::= SEQUENCE {
        isupport        [0] KerberosVSupportNego
                                -- Version of Kerberos V supported by
                                -- the target realm.
   }

   Err-get-realm-krb5-support   ::= Extensible-NULL

   -- Get s2kparams
   Req-get-s2kparams            ::= Extensible-NULL

   Rep-get-s2kparams            ::= SEQUENCE {
        help-text       [0] UTF8String OPTIONAL,
        s2kparams       [1] SEQUENCE OF Etype-Info-Entry
   }

   Err-get-s2kparams            ::= Extensible-NULL
```

```
END
```

6. Security Considerations

Implementors and site administrators should note that the redundancy of UTF-8 encodings of passwords varies by language. Password quality policies SHOULD, therefore, take this into account when estimating the amount of redundancy and entropy in a proposed new password.

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 [RFC4120]). Clients SHOULD NOT negotiate support for legacy major versions of this protocol unless configured to do so.

This protocol does not have Perfect Forward Security (PFS). As a result, any passive network snooper watching password/key changing operations who has stolen a principal's password or long-term keys can find out what the new ones are.

7. Normative References

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

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3244] Swift, M., Trostle, J., and J. Brezak, "Microsoft Windows 2000 Kerberos Change Password and Set Password Protocols", RFC 3244, February 2002.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[RFC4646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 4646, September 2006.

[RFC4647] Phillips, A. and M. Davis, "Matching of Language Tags", BCP 47, RFC 4647, September 2006.

Author's Address

Nicolas Williams
Sun Microsystems
5300 Riata Trace Ct
Austin, TX 78727
US

Email: Nicolas.Williams@sun.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).