Kerberos User Guide

Release 1.15.1

MIT

1 Password management **1**
1.1 Changing your password . . . 1
1.2 Granting access to your account . . . 2
1.3 Password quality verification . . . 2

2 Ticket management **3**
2.1 Kerberos ticket properties . . . 3
2.2 Obtaining tickets with kinit . . . 4
2.3 Viewing tickets with klist . . . 5
2.4 Destroying tickets with kdestroy . . . 7

3 User config files **9**
3.1 .k5login . . . 9
3.2 .k5identity . . . 10

4 User commands **11**
4.1 kdestroy . . . 11
4.2 kinit . . . 12
4.3 klist . . . 14
4.4 kpasswd . . . 16
4.5 krb5-config . . . 16
4.6 ksu . . . 17
4.7 kswitch . . . 22
4.8 kvno . . . 22
4.9 sclient . . . 23

CHAPTER

ONE

PASSWORD MANAGEMENT

Your password is the only way Kerberos has of verifying your identity. If someone finds out your password, that person can masquerade as you—send email that comes from you, read, edit, or delete your files, or log into other hosts as you—and no one will be able to tell the difference. For this reason, it is important that you choose a good password, and keep it secret. If you need to give access to your account to someone else, you can do so through Kerberos (see *Granting access to your account*). You should never tell your password to anyone, including your system administrator, for any reason. You should change your password frequently, particularly any time you think someone may have found out what it is.

1.1 Changing your password

To change your Kerberos password, use the *kpasswd* command. It will ask you for your old password (to prevent someone else from walking up to your computer when you're not there and changing your password), and then prompt you for the new one twice. (The reason you have to type it twice is to make sure you have typed it correctly.) For example, user `david` would do the following:

```
shell% kpasswd
Password for david:    <- Type your old password.
Enter new password:    <- Type your new password.
Enter it again:  <- Type the new password again.
Password changed.
shell%
```

If `david` typed the incorrect old password, he would get the following message:

```
shell% kpasswd
Password for david:  <- Type the incorrect old password.
kpasswd: Password incorrect while getting initial ticket
shell%
```

If you make a mistake and don't type the new password the same way twice, kpasswd will ask you to try again:

```
shell% kpasswd
Password for david:  <- Type the old password.
Enter new password:  <- Type the new password.
Enter it again: <- Type a different new password.
kpasswd: Password mismatch while reading password
shell%
```

Once you change your password, it takes some time for the change to propagate through the system. Depending on how your system is set up, this might be anywhere from a few minutes to an hour or more. If you need to get new

Kerberos tickets shortly after changing your password, try the new password. If the new password doesn't work, try again using the old one.

1.2 Granting access to your account

If you need to give someone access to log into your account, you can do so through Kerberos, without telling the person your password. Simply create a file called *.k5login* in your home directory. This file should contain the Kerberos principal of each person to whom you wish to give access. Each principal must be on a separate line. Here is a sample .k5login file:

```
jennifer@ATHENA.MIT.EDU
david@EXAMPLE.COM
```

This file would allow the users `jennifer` and `david` to use your user ID, provided that they had Kerberos tickets in their respective realms. If you will be logging into other hosts across a network, you will want to include your own Kerberos principal in your .k5login file on each of these hosts.

Using a .k5login file is much safer than giving out your password, because:

- You can take access away any time simply by removing the principal from your .k5login file.
- Although the user has full access to your account on one particular host (or set of hosts if your .k5login file is shared, e.g., over NFS), that user does not inherit your network privileges.
- Kerberos keeps a log of who obtains tickets, so a system administrator could find out, if necessary, who was capable of using your user ID at a particular time.

One common application is to have a .k5login file in root's home directory, giving root access to that machine to the Kerberos principals listed. This allows system administrators to allow users to become root locally, or to log in remotely as root, without their having to give out the root password, and without anyone having to type the root password over the network.

1.3 Password quality verification

TODO

CHAPTER

TWO

TICKET MANAGEMENT

On many systems, Kerberos is built into the login program, and you get tickets automatically when you log in. Other programs, such as ssh, can forward copies of your tickets to a remote host. Most of these programs also automatically destroy your tickets when they exit. However, MIT recommends that you explicitly destroy your Kerberos tickets when you are through with them, just to be sure. One way to help ensure that this happens is to add the *kdestroy* command to your .logout file. Additionally, if you are going to be away from your machine and are concerned about an intruder using your permissions, it is safest to either destroy all copies of your tickets, or use a screensaver that locks the screen.

2.1 Kerberos ticket properties

There are various properties that Kerberos tickets can have:

If a ticket is **forwardable**, then the KDC can issue a new ticket (with a different network address, if necessary) based on the forwardable ticket. This allows for authentication forwarding without requiring a password to be typed in again. For example, if a user with a forwardable TGT logs into a remote system, the KDC could issue a new TGT for that user with the network address of the remote system, allowing authentication on that host to work as though the user were logged in locally.

When the KDC creates a new ticket based on a forwardable ticket, it sets the **forwarded** flag on that new ticket. Any tickets that are created based on a ticket with the forwarded flag set will also have their forwarded flags set.

A **proxiable** ticket is similar to a forwardable ticket in that it allows a service to take on the identity of the client. Unlike a forwardable ticket, however, a proxiable ticket is only issued for specific services. In other words, a ticket-granting ticket cannot be issued based on a ticket that is proxiable but not forwardable.

A **proxy** ticket is one that was issued based on a proxiable ticket.

A **postdated** ticket is issued with the invalid flag set. After the starting time listed on the ticket, it can be presented to the KDC to obtain valid tickets.

Ticket-granting tickets with the **postdateable** flag set can be used to obtain postdated service tickets.

Renewable tickets can be used to obtain new session keys without the user entering their password again. A renewable ticket has two expiration times. The first is the time at which this particular ticket expires. The second is the latest possible expiration time for any ticket issued based on this renewable ticket.

A ticket with the **initial flag** set was issued based on the authentication protocol, and not on a ticket-granting ticket. Application servers that wish to ensure that the user's key has been recently presented for verification could specify that this flag must be set to accept the ticket.

An **invalid** ticket must be rejected by application servers. Postdated tickets are usually issued with this flag set, and must be validated by the KDC before they can be used.

A **preauthenticated** ticket is one that was only issued after the client requesting the ticket had authenticated itself to the KDC.

The **hardware authentication** flag is set on a ticket which required the use of hardware for authentication. The hardware is expected to be possessed only by the client which requested the tickets.

If a ticket has the **transit policy** checked flag set, then the KDC that issued this ticket implements the transited-realm check policy and checked the transited-realms list on the ticket. The transited-realms list contains a list of all intermediate realms between the realm of the KDC that issued the first ticket and that of the one that issued the current ticket. If this flag is not set, then the application server must check the transited realms itself or else reject the ticket.

The **okay as delegate** flag indicates that the server specified in the ticket is suitable as a delegate as determined by the policy of that realm. Some client applications may use this flag to decide whether to forward tickets to a remote host, although many applications do not honor it.

An **anonymous** ticket is one in which the named principal is a generic principal for that realm; it does not actually specify the individual that will be using the ticket. This ticket is meant only to securely distribute a session key.

2.2 Obtaining tickets with kinit

If your site has integrated Kerberos V5 with the login system, you will get Kerberos tickets automatically when you log in. Otherwise, you may need to explicitly obtain your Kerberos tickets, using the *kinit* program. Similarly, if your Kerberos tickets expire, use the kinit program to obtain new ones.

To use the kinit program, simply type `kinit` and then type your password at the prompt. For example, Jennifer (whose username is `jennifer`) works for Bleep, Inc. (a fictitious company with the domain name mit.edu and the Kerberos realm ATHENA.MIT.EDU). She would type:

```
shell% kinit
Password for jennifer@ATHENA.MIT.EDU: <-- [Type jennifer's password here.]
shell%
```

If you type your password incorrectly, kinit will give you the following error message:

```
shell% kinit
Password for jennifer@ATHENA.MIT.EDU: <-- [Type the wrong password here.]
kinit: Password incorrect
shell%
```

and you won't get Kerberos tickets.

By default, kinit assumes you want tickets for your own username in your default realm. Suppose Jennifer's friend David is visiting, and he wants to borrow a window to check his mail. David needs to get tickets for himself in his own realm, EXAMPLE.COM. He would type:

```
shell% kinit david@EXAMPLE.COM
Password for david@EXAMPLE.COM: <-- [Type david's password here.]
shell%
```

David would then have tickets which he could use to log onto his own machine. Note that he typed his password locally on Jennifer's machine, but it never went over the network. Kerberos on the local host performed the authentication to the KDC in the other realm.

If you want to be able to forward your tickets to another host, you need to request forwardable tickets. You do this by specifying the **-f** option:

```
shell% kinit -f
Password for jennifer@ATHENA.MIT.EDU: <-- [Type your password here.]
shell%
```

Note that kinit does not tell you that it obtained forwardable tickets; you can verify this using the *klist* command (see *Viewing tickets with klist*).

Normally, your tickets are good for your system's default ticket lifetime, which is ten hours on many systems. You can specify a different ticket lifetime with the **-l** option. Add the letter **s** to the value for seconds, **m** for minutes, **h** for hours, or **d** for days. For example, to obtain forwardable tickets for `david@EXAMPLE.COM` that would be good for three hours, you would type:

```
shell% kinit -f -l 3h david@EXAMPLE.COM
Password for david@EXAMPLE.COM: <-- [Type david's password here.]
shell%
```

Note: You cannot mix units; specifying a lifetime of 3h30m would result in an error. Note also that most systems specify a maximum ticket lifetime. If you request a longer ticket lifetime, it will be automatically truncated to the maximum lifetime.

2.3 Viewing tickets with klist

The *klist* command shows your tickets. When you first obtain tickets, you will have only the ticket-granting ticket. The listing would look like this:

```
shell% klist
Ticket cache: /tmp/krb5cc_ttypa
Default principal: jennifer@ATHENA.MIT.EDU

Valid starting     Expires            Service principal
06/07/04 19:49:21  06/08/04 05:49:19  krbtgt/ATHENA.MIT.EDU@ATHENA.MIT.EDU
shell%
```

The ticket cache is the location of your ticket file. In the above example, this file is named `/tmp/krb5cc_ttypa`. The default principal is your Kerberos principal.

The "valid starting" and "expires" fields describe the period of time during which the ticket is valid. The "service principal" describes each ticket. The ticket-granting ticket has a first component `krbtgt`, and a second component which is the realm name.

Now, if `jennifer` connected to the machine `daffodil.mit.edu`, and then typed "klist" again, she would have gotten the following result:

```
shell% klist
Ticket cache: /tmp/krb5cc_ttypa
Default principal: jennifer@ATHENA.MIT.EDU

Valid starting     Expires            Service principal
06/07/04 19:49:21  06/08/04 05:49:19  krbtgt/ATHENA.MIT.EDU@ATHENA.MIT.EDU
06/07/04 20:22:30  06/08/04 05:49:19  host/daffodil.mit.edu@ATHENA.MIT.EDU
shell%
```

Here's what happened: when `jennifer` used ssh to connect to the host `daffodil.mit.edu`, the ssh program presented her ticket-granting ticket to the KDC and requested a host ticket for the host `daffodil.mit.edu`. The

KDC sent the host ticket, which ssh then presented to the host `daffodil.mit.edu`, and she was allowed to log in without typing her password.

Suppose your Kerberos tickets allow you to log into a host in another domain, such as `trillium.example.com`, which is also in another Kerberos realm, `EXAMPLE.COM`. If you ssh to this host, you will receive a ticket-granting ticket for the realm `EXAMPLE.COM`, plus the new host ticket for `trillium.example.com`. klist will now show:

```
shell% klist
Ticket cache: /tmp/krb5cc_ttypa
Default principal: jennifer@ATHENA.MIT.EDU

Valid starting     Expires            Service principal
06/07/04 19:49:21  06/08/04 05:49:19  krbtgt/ATHENA.MIT.EDU@ATHENA.MIT.EDU
06/07/04 20:22:30  06/08/04 05:49:19  host/daffodil.mit.edu@ATHENA.MIT.EDU
06/07/04 20:24:18  06/08/04 05:49:19  krbtgt/EXAMPLE.COM@ATHENA.MIT.EDU
06/07/04 20:24:18  06/08/04 05:49:19  host/trillium.example.com@EXAMPLE.COM
shell%
```

Depending on your host's and realm's configuration, you may also see a ticket with the service principal `host/trillium.example.com@`. If so, this means that your host did not know what realm trillium.example.com is in, so it asked the `ATHENA.MIT.EDU` KDC for a referral. The next time you connect to `trillium.example.com`, the odd-looking entry will be used to avoid needing to ask for a referral again.

You can use the **-f** option to view the flags that apply to your tickets. The flags are:

F	Forwardable
f	forwarded
P	Proxiable
p	proxy
D	postDateable
d	postdated
R	Renewable
I	Initial
i	invalid
H	Hardware authenticated
A	preAuthenticated
T	Transit policy checked
O	Okay as delegate
a	anonymous

Here is a sample listing. In this example, the user *jennifer* obtained her initial tickets (**I**), which are forwardable (**F**) and postdated (**d**) but not yet validated (**i**):

```
shell% klist -f
Ticket cache: /tmp/krb5cc_320
Default principal: jennifer@ATHENA.MIT.EDU

Valid starting      Expires             Service principal
31/07/05 19:06:25  31/07/05 19:16:25  krbtgt/ATHENA.MIT.EDU@ATHENA.MIT.EDU
        Flags: FdiI
shell%
```

In the following example, the user *david*'s tickets were forwarded (**f**) to this host from another host. The tickets are reforwardable (**F**):

```
shell% klist -f
Ticket cache: /tmp/krb5cc_p11795
Default principal: david@EXAMPLE.COM
```

```
Valid starting     Expires            Service principal
07/31/05 11:52:29  07/31/05 21:11:23  krbtgt/EXAMPLE.COM@EXAMPLE.COM
        Flags: Ff
07/31/05 12:03:48  07/31/05 21:11:23  host/trillium.example.com@EXAMPLE.COM
        Flags: Ff
shell%
```

2.4 Destroying tickets with kdestroy

Your Kerberos tickets are proof that you are indeed yourself, and tickets could be stolen if someone gains access to a computer where they are stored. If this happens, the person who has them can masquerade as you until they expire. For this reason, you should destroy your Kerberos tickets when you are away from your computer.

Destroying your tickets is easy. Simply type kdestroy:

```
shell% kdestroy
shell%
```

If *kdestroy* fails to destroy your tickets, it will beep and give an error message. For example, if kdestroy can't find any tickets to destroy, it will give the following message:

```
shell% kdestroy
kdestroy: No credentials cache file found while destroying cache
shell%
```

CHAPTER

THREE

USER CONFIG FILES

The following files in your home directory can be used to control the behavior of Kerberos as it applies to your account (unless they have been disabled by your host's configuration):

3.1 .k5login

3.1.1 DESCRIPTION

The .k5login file, which resides in a user's home directory, contains a list of the Kerberos principals. Anyone with valid tickets for a principal in the file is allowed host access with the UID of the user in whose home directory the file resides. One common use is to place a .k5login file in root's home directory, thereby granting system administrators remote root access to the host via Kerberos.

3.1.2 EXAMPLES

Suppose the user `alice` had a .k5login file in her home directory containing just the following line:

```
bob@FOOBAR.ORG
```

This would allow `bob` to use Kerberos network applications, such as ssh(1), to access `alice`'s account, using `bob`'s Kerberos tickets. In a default configuration (with **k5login_authoritative** set to true in krb5.conf(5)), this .k5login file would not let `alice` use those network applications to access her account, since she is not listed! With no .k5login file, or with **k5login_authoritative** set to false, a default rule would permit the principal `alice` in the machine's default realm to access the `alice` account.

Let us further suppose that `alice` is a system administrator. Alice and the other system administrators would have their principals in root's .k5login file on each host:

```
alice@BLEEP.COM

joeadmin/root@BLEEP.COM
```

This would allow either system administrator to log in to these hosts using their Kerberos tickets instead of having to type the root password. Note that because `bob` retains the Kerberos tickets for his own principal, `bob@FOOBAR.ORG`, he would not have any of the privileges that require `alice`'s tickets, such as root access to any of the site's hosts, or the ability to change `alice`'s password.

3.1.3 SEE ALSO

kerberos(1)

3.2 .k5identity

3.2.1 DESCRIPTION

The .k5identity file, which resides in a user's home directory, contains a list of rules for selecting a client principals based on the server being accessed. These rules are used to choose a credential cache within the cache collection when possible.

Blank lines and lines beginning with # are ignored. Each line has the form:

principal field=value ...

If the server principal meets all of the field constraints, then principal is chosen as the client principal. The following fields are recognized:

realm If the realm of the server principal is known, it is matched against *value*, which may be a pattern using shell wildcards. For host-based server principals, the realm will generally only be known if there is a domain_realm section in krb5.conf(5) with a mapping for the hostname.

service If the server principal is a host-based principal, its service component is matched against *value*, which may be a pattern using shell wildcards.

host If the server principal is a host-based principal, its hostname component is converted to lower case and matched against *value*, which may be a pattern using shell wildcards.

If the server principal matches the constraints of multiple lines in the .k5identity file, the principal from the first matching line is used. If no line matches, credentials will be selected some other way, such as the realm heuristic or the current primary cache.

3.2.2 EXAMPLE

The following example .k5identity file selects the client principal `alice@KRBTEST.COM` if the server principal is within that realm, the principal `alice/root@EXAMPLE.COM` if the server host is within a servers subdomain, and the principal `alice/mail@EXAMPLE.COM` when accessing the IMAP service on `mail.example.com`:

```
alice@KRBTEST.COM       realm=KRBTEST.COM
alice/root@EXAMPLE.COM  host=*.servers.example.com
alice/mail@EXAMPLE.COM  host=mail.example.com service=imap
```

3.2.3 SEE ALSO

kerberos(1), krb5.conf(5)

CHAPTER

FOUR

USER COMMANDS

4.1 kdestroy

4.1.1 SYNOPSIS

kdestroy [**-A**] [**-q**] [**-c** *cache_name*]

4.1.2 DESCRIPTION

The kdestroy utility destroys the user's active Kerberos authorization tickets by overwriting and deleting the credentials cache that contains them. If the credentials cache is not specified, the default credentials cache is destroyed.

4.1.3 OPTIONS

-A Destroys all caches in the collection, if a cache collection is available.

-q Run quietly. Normally kdestroy beeps if it fails to destroy the user's tickets. The **-q** flag suppresses this behavior.

-c *cache_name* Use *cache_name* as the credentials (ticket) cache name and location; if this option is not used, the default cache name and location are used.

The default credentials cache may vary between systems. If the **KRB5CCNAME** environment variable is set, its value is used to name the default ticket cache.

4.1.4 NOTE

Most installations recommend that you place the kdestroy command in your .logout file, so that your tickets are destroyed automatically when you log out.

4.1.5 ENVIRONMENT

kdestroy uses the following environment variable:

KRB5CCNAME Location of the default Kerberos 5 credentials (ticket) cache, in the form *type*:*residual*. If no *type* prefix is present, the **FILE** type is assumed. The type of the default cache may determine the availability of a cache collection; for instance, a default cache of type **DIR** causes caches within the directory to be present in the collection.

4.1.6 FILES

FILE:/tmp/krb5cc_%{uid} Default location of Kerberos 5 credentials cache

4.1.7 SEE ALSO

kinit, *klist*

4.2 kinit

4.2.1 SYNOPSIS

kinit [**-V**] [**-l** *lifetime*] [**-s** *start_time*] [**-r** *renewable_life*] [**-p** | **-P**] [**-f** | **-F**] [**-a**] [**-A**] [**-C**] [**-E**] [**-v**] [**-R**] [**-k** [**-t** *keytab_file*]] [**-c** *cache_name*] [**-n**] [**-S** *service_name*] [**-I** *input_ccache*] [**-T** *armor_ccache*] [**-X** *attribute*[=*value*]] [*principal*]

4.2.2 DESCRIPTION

kinit obtains and caches an initial ticket-granting ticket for *principal*. If *principal* is absent, kinit chooses an appropriate principal name based on existing credential cache contents or the local username of the user invoking kinit. Some options modify the choice of principal name.

4.2.3 OPTIONS

-V display verbose output.

-l *lifetime* (duration string.) Requests a ticket with the lifetime *lifetime*.

For example, `kinit -l 5:30` or `kinit -l 5h30m`.

If the **-l** option is not specified, the default ticket lifetime (configured by each site) is used. Specifying a ticket lifetime longer than the maximum ticket lifetime (configured by each site) will not override the configured maximum ticket lifetime.

-s *start_time* (duration string.) Requests a postdated ticket. Postdated tickets are issued with the **invalid** flag set, and need to be resubmitted to the KDC for validation before use.

start_time specifies the duration of the delay before the ticket can become valid.

-r *renewable_life* (duration string.) Requests renewable tickets, with a total lifetime of *renewable_life*.

-f requests forwardable tickets.

-F requests non-forwardable tickets.

-p requests proxiable tickets.

-P requests non-proxiable tickets.

-a requests tickets restricted to the host's local address[es].

-A requests tickets not restricted by address.

-C requests canonicalization of the principal name, and allows the KDC to reply with a different client principal from the one requested.

-E treats the principal name as an enterprise name (implies the **-C** option).

-v requests that the ticket-granting ticket in the cache (with the **invalid** flag set) be passed to the KDC for validation. If the ticket is within its requested time range, the cache is replaced with the validated ticket.

-R requests renewal of the ticket-granting ticket. Note that an expired ticket cannot be renewed, even if the ticket is still within its renewable life.

Note that renewable tickets that have expired as reported by *klist* may sometimes be renewed using this option, because the KDC applies a grace period to account for client-KDC clock skew. See krb5.conf(5) **clockskew** setting.

-k [-i | -t *keytab_file*] requests a ticket, obtained from a key in the local host's keytab. The location of the keytab may be specified with the **-t** *keytab_file* option, or with the **-i** option to specify the use of the default client keytab; otherwise the default keytab will be used. By default, a host ticket for the local host is requested, but any principal may be specified. On a KDC, the special keytab location `KDB:` can be used to indicate that kinit should open the KDC database and look up the key directly. This permits an administrator to obtain tickets as any principal that supports authentication based on the key.

-n Requests anonymous processing. Two types of anonymous principals are supported.

For fully anonymous Kerberos, configure pkinit on the KDC and configure **pkinit_anchors** in the client's krb5.conf(5). Then use the **-n** option with a principal of the form `@REALM` (an empty principal name followed by the at-sign and a realm name). If permitted by the KDC, an anonymous ticket will be returned.

A second form of anonymous tickets is supported; these realm-exposed tickets hide the identity of the client but not the client's realm. For this mode, use `kinit -n` with a normal principal name. If supported by the KDC, the principal (but not realm) will be replaced by the anonymous principal.

As of release 1.8, the MIT Kerberos KDC only supports fully anonymous operation.

-I *input_ccache*

Specifies the name of a credentials cache that already contains a ticket. When obtaining that ticket, if information about how that ticket was obtained was also stored to the cache, that information will be used to affect how new credentials are obtained, including preselecting the same methods of authenticating to the KDC.

-T *armor_ccache* Specifies the name of a credentials cache that already contains a ticket. If supported by the KDC, this cache will be used to armor the request, preventing offline dictionary attacks and allowing the use of additional preauthentication mechanisms. Armoring also makes sure that the response from the KDC is not modified in transit.

-c *cache_name* use *cache_name* as the Kerberos 5 credentials (ticket) cache location. If this option is not used, the default cache location is used.

The default cache location may vary between systems. If the **KRB5CCNAME** environment variable is set, its value is used to locate the default cache. If a principal name is specified and the type of the default cache supports a collection (such as the DIR type), an existing cache containing credentials for the principal is selected or a new one is created and becomes the new primary cache. Otherwise, any existing contents of the default cache are destroyed by kinit.

-S *service_name* specify an alternate service name to use when getting initial tickets.

-X *attribute*[=*value*] specify a pre-authentication *attribute* and *value* to be interpreted by pre-authentication modules. The acceptable attribute and value values vary from module to module. This option may be specified multiple times to specify multiple attributes. If no value is specified, it is assumed to be "yes".

The following attributes are recognized by the PKINIT pre-authentication mechanism:

X509_user_identity=*value* specify where to find user's X509 identity information

X509_anchors=*value* specify where to find trusted X509 anchor information

flag_RSA_PROTOCOL[=yes] specify use of RSA, rather than the default Diffie-Hellman protocol

4.2.4 ENVIRONMENT

kinit uses the following environment variables:

KRB5CCNAME Location of the default Kerberos 5 credentials cache, in the form *type*:*residual*. If no *type* prefix is present, the **FILE** type is assumed. The type of the default cache may determine the availability of a cache collection; for instance, a default cache of type **DIR** causes caches within the directory to be present in the collection.

4.2.5 FILES

FILE:/tmp/krb5cc_%{uid} default location of Kerberos 5 credentials cache

FILE:/etc/krb5.keytab default location for the local host's keytab.

4.2.6 SEE ALSO

klist, *kdestroy*, kerberos(1)

4.3 klist

4.3.1 SYNOPSIS

klist [**-e**] [[**-c**] [**-l**] [**-A**] [**-f**] [**-s**] [**-a** [**-n**]]] [**-C**] [**-k** [**-t**] [**-K**]] [**-V**] [*cache_name*|*keytab_name*]

4.3.2 DESCRIPTION

klist lists the Kerberos principal and Kerberos tickets held in a credentials cache, or the keys held in a keytab file.

4.3.3 OPTIONS

-e Displays the encryption types of the session key and the ticket for each credential in the credential cache, or each key in the keytab file.

-l If a cache collection is available, displays a table summarizing the caches present in the collection.

-A If a cache collection is available, displays the contents of all of the caches in the collection.

-c List tickets held in a credentials cache. This is the default if neither **-c** nor **-k** is specified.

-f Shows the flags present in the credentials, using the following abbreviations:

```
F    Forwardable
f    forwarded
P    Proxiable
p    proxy
D    postDateable
d    postdated
R    Renewable
```

```
I    Initial
i    invalid
H    Hardware authenticated
A    preAuthenticated
T    Transit policy checked
O    Okay as delegate
a    anonymous
```

-s Causes klist to run silently (produce no output). klist will exit with status 1 if the credentials cache cannot be read or is expired, and with status 0 otherwise.

-a Display list of addresses in credentials.

-n Show numeric addresses instead of reverse-resolving addresses.

-C List configuration data that has been stored in the credentials cache when klist encounters it. By default, configuration data is not listed.

-k List keys held in a keytab file.

-i In combination with **-k**, defaults to using the default client keytab instead of the default acceptor keytab, if no name is given.

-t Display the time entry timestamps for each keytab entry in the keytab file.

-K Display the value of the encryption key in each keytab entry in the keytab file.

-V Display the Kerberos version number and exit.

If *cache_name* or *keytab_name* is not specified, klist will display the credentials in the default credentials cache or keytab file as appropriate. If the **KRB5CCNAME** environment variable is set, its value is used to locate the default ticket cache.

4.3.4 ENVIRONMENT

klist uses the following environment variable:

KRB5CCNAME Location of the default Kerberos 5 credentials (ticket) cache, in the form *type*:*residual*. If no *type* prefix is present, the **FILE** type is assumed. The type of the default cache may determine the availability of a cache collection; for instance, a default cache of type **DIR** causes caches within the directory to be present in the collection.

4.3.5 FILES

FILE:/tmp/krb5cc_%{uid} Default location of Kerberos 5 credentials cache

FILE:/etc/krb5.keytab Default location for the local host's keytab file.

4.3.6 SEE ALSO

kinit, *kdestroy*

4.4 kpasswd

4.4.1 SYNOPSIS

kpasswd [*principal*]

4.4.2 DESCRIPTION

The kpasswd command is used to change a Kerberos principal's password. kpasswd first prompts for the current Kerberos password, then prompts the user twice for the new password, and the password is changed.

If the principal is governed by a policy that specifies the length and/or number of character classes required in the new password, the new password must conform to the policy. (The five character classes are lower case, upper case, numbers, punctuation, and all other characters.)

4.4.3 OPTIONS

principal Change the password for the Kerberos principal principal. Otherwise, kpasswd uses the principal name from an existing ccache if there is one; if not, the principal is derived from the identity of the user invoking the kpasswd command.

4.4.4 SEE ALSO

kadmin(1), kadmind(8)

4.5 krb5-config

4.5.1 SYNOPSIS

krb5-config [**--help** | **--all** | **--version** | **--vendor** | **--prefix** | **--exec-prefix** | **--defccname** | **--defktname** | **--defcktname** | **--cflags** | **--libs** [*libraries*]]

4.5.2 DESCRIPTION

krb5-config tells the application programmer what flags to use to compile and link programs against the installed Kerberos libraries.

4.5.3 OPTIONS

--help prints a usage message. This is the default behavior when no options are specified.

--all prints the version, vendor, prefix, and exec-prefix.

--version prints the version number of the Kerberos installation.

--vendor prints the name of the vendor of the Kerberos installation.

--prefix prints the prefix for which the Kerberos installation was built.

--exec-prefix prints the prefix for executables for which the Kerberos installation was built.

--defccname prints the built-in default credentials cache location.

--defktname prints the built-in default keytab location.

--defcktname prints the built-in default client (initiator) keytab location.

--cflags prints the compilation flags used to build the Kerberos installation.

--libs [*library*] prints the compiler options needed to link against *library*. Allowed values for *library* are:

krb5	Kerberos 5 applications (default)
gssapi	GSSAPI applications with Kerberos 5 bindings
kadm-client	Kadmin client
kadm-server	Kadmin server
kdb	Applications that access the Kerberos database

4.5.4 EXAMPLES

krb5-config is particularly useful for compiling against a Kerberos installation that was installed in a non-standard location. For example, a Kerberos installation that is installed in `/opt/krb5/` but uses libraries in `/usr/local/lib/` for text localization would produce the following output:

```
shell% krb5-config --libs krb5
-L/opt/krb5/lib -Wl,-rpath -Wl,/opt/krb5/lib -L/usr/local/lib -lkrb5 -lk5crypto -lcom_
↪err
```

4.5.5 SEE ALSO

kerberos(1), cc(1)

4.6 ksu

4.6.1 SYNOPSIS

ksu [*target_user*] [**-n** *target_principal_name*] [**-c** *source_cache_name*] [**-k**] [**-r** time] [**-pf**] [**-l** *lifetime*] [**-z | Z**] [**-q**] [**-e** *command* [args ...]] [**-a** [args ...]]

4.6.2 REQUIREMENTS

Must have Kerberos version 5 installed to compile ksu. Must have a Kerberos version 5 server running to use ksu.

4.6.3 DESCRIPTION

ksu is a Kerberized version of the su program that has two missions: one is to securely change the real and effective user ID to that of the target user, and the other is to create a new security context.

Note: For the sake of clarity, all references to and attributes of the user invoking the program will start with "source" (e.g., "source user", "source cache", etc.).

Likewise, all references to and attributes of the target account will start with "target".

4.6.4 AUTHENTICATION

To fulfill the first mission, ksu operates in two phases: authentication and authorization. Resolving the target principal name is the first step in authentication. The user can either specify his principal name with the **-n** option (e.g., `-n jqpublic@USC.EDU`) or a default principal name will be assigned using a heuristic described in the OPTIONS section (see **-n** option). The target user name must be the first argument to ksu; if not specified root is the default. If **.** is specified then the target user will be the source user (e.g., `ksu .`). If the source user is root or the target user is the source user, no authentication or authorization takes place. Otherwise, ksu looks for an appropriate Kerberos ticket in the source cache.

The ticket can either be for the end-server or a ticket granting ticket (TGT) for the target principal's realm. If the ticket for the end-server is already in the cache, it's decrypted and verified. If it's not in the cache but the TGT is, the TGT is used to obtain the ticket for the end-server. The end-server ticket is then verified. If neither ticket is in the cache, but ksu is compiled with the **GET_TGT_VIA_PASSWD** define, the user will be prompted for a Kerberos password which will then be used to get a TGT. If the user is logged in remotely and does not have a secure channel, the password may be exposed. If neither ticket is in the cache and **GET_TGT_VIA_PASSWD** is not defined, authentication fails.

4.6.5 AUTHORIZATION

This section describes authorization of the source user when ksu is invoked without the **-e** option. For a description of the **-e** option, see the OPTIONS section.

Upon successful authentication, ksu checks whether the target principal is authorized to access the target account. In the target user's home directory, ksu attempts to access two authorization files: *.k5login* and .k5users. In the .k5login file each line contains the name of a principal that is authorized to access the account.

For example:

```
jqpublic@USC.EDU
jqpublic/secure@USC.EDU
jqpublic/admin@USC.EDU
```

The format of .k5users is the same, except the principal name may be followed by a list of commands that the principal is authorized to execute (see the **-e** option in the OPTIONS section for details).

Thus if the target principal name is found in the .k5login file the source user is authorized to access the target account. Otherwise ksu looks in the .k5users file. If the target principal name is found without any trailing commands or followed only by `*` then the source user is authorized. If either .k5login or .k5users exist but an appropriate entry for the target principal does not exist then access is denied. If neither file exists then the principal will be granted access to the account according to the aname->lname mapping rules. Otherwise, authorization fails.

4.6.6 EXECUTION OF THE TARGET SHELL

Upon successful authentication and authorization, ksu proceeds in a similar fashion to su. The environment is unmodified with the exception of USER, HOME and SHELL variables. If the target user is not root, USER gets set to the target user name. Otherwise USER remains unchanged. Both HOME and SHELL are set to the target login's default values. In addition, the environment variable **KRB5CCNAME** gets set to the name of the target cache. The real and effective user ID are changed to that of the target user. The target user's shell is then invoked (the shell name is specified in the password file). Upon termination of the shell, ksu deletes the target cache (unless ksu is invoked with the **-k** option). This is implemented by first doing a fork and then an exec, instead of just exec, as done by su.

4.6.7 CREATING A NEW SECURITY CONTEXT

ksu can be used to create a new security context for the target program (either the target shell, or command specified via the **-e** option). The target program inherits a set of credentials from the source user. By default, this set includes all of the credentials in the source cache plus any additional credentials obtained during authentication. The source user is able to limit the credentials in this set by using **-z** or **-Z** option. **-z** restricts the copy of tickets from the source cache to the target cache to only the tickets where client == the target principal name. The **-Z** option provides the target user with a fresh target cache (no creds in the cache). Note that for security reasons, when the source user is root and target user is non-root, **-z** option is the default mode of operation.

While no authentication takes place if the source user is root or is the same as the target user, additional tickets can still be obtained for the target cache. If **-n** is specified and no credentials can be copied to the target cache, the source user is prompted for a Kerberos password (unless **-Z** specified or **GET_TGT_VIA_PASSWD** is undefined). If successful, a TGT is obtained from the Kerberos server and stored in the target cache. Otherwise, if a password is not provided (user hit return) ksu continues in a normal mode of operation (the target cache will not contain the desired TGT). If the wrong password is typed in, ksu fails.

Note: During authentication, only the tickets that could be obtained without providing a password are cached in in the source cache.

4.6.8 OPTIONS

-n *target_principal_name* Specify a Kerberos target principal name. Used in authentication and authorization phases of ksu.

If ksu is invoked without **-n**, a default principal name is assigned via the following heuristic:

- Case 1: source user is non-root.

  If the target user is the source user the default principal name is set to the default principal of the source cache. If the cache does not exist then the default principal name is set to `target_user@local_realm`. If the source and target users are different and neither `~target_user/.k5users` nor `~target_user/.k5login` exist then the default principal name is `target_user_login_name@local_realm`. Otherwise, starting with the first principal listed below, ksu checks if the principal is authorized to access the target account and whether there is a legitimate ticket for that principal in the source cache. If both conditions are met that principal becomes the default target principal, otherwise go to the next principal.

  1. default principal of the source cache
  2. target_user@local_realm
  3. source_user@local_realm

  If a-c fails try any principal for which there is a ticket in the source cache and that is authorized to access the target account. If that fails select the first principal that is authorized to access the target account from the above list. If none are authorized and ksu is configured with **PRINC_LOOK_AHEAD** turned on, select the default principal as follows:

  For each candidate in the above list, select an authorized principal that has the same realm name and first part of the principal name equal to the prefix of the candidate. For example if candidate a) is `jqpublic@ISI.EDU` and `jqpublic/secure@ISI.EDU` is authorized to access the target account then the default principal is set to `jqpublic/secure@ISI.EDU`.

- Case 2: source user is root.

If the target user is non-root then the default principal name is `target_user@local_realm`. Else, if the source cache exists the default principal name is set to the default principal of the source cache. If the source cache does not exist, default principal name is set to `root\@local_realm`.

-c *source_cache_name*

Specify source cache name (e.g., `-c FILE:/tmp/my_cache`). If **-c** option is not used then the name is obtained from **KRB5CCNAME** environment variable. If **KRB5CCNAME** is not defined the source cache name is set to `krb5cc_<source uid>`. The target cache name is automatically set to `krb5cc_<target uid>.(gen_sym())`, where gen_sym generates a new number such that the resulting cache does not already exist. For example:

```
krb5cc_1984.2
```

-k Do not delete the target cache upon termination of the target shell or a command (**-e** command). Without **-k**, ksu deletes the target cache.

-z Restrict the copy of tickets from the source cache to the target cache to only the tickets where client == the target principal name. Use the **-n** option if you want the tickets for other then the default principal. Note that the **-z** option is mutually exclusive with the **-Z** option.

-Z Don't copy any tickets from the source cache to the target cache. Just create a fresh target cache, where the default principal name of the cache is initialized to the target principal name. Note that the **-Z** option is mutually exclusive with the **-z** option.

-q Suppress the printing of status messages.

Ticket granting ticket options:

-l ***lifetime*** **-r** ***time*** **-pf** The ticket granting ticket options only apply to the case where there are no appropriate tickets in the cache to authenticate the source user. In this case if ksu is configured to prompt users for a Kerberos password (**GET_TGT_VIA_PASSWD** is defined), the ticket granting ticket options that are specified will be used when getting a ticket granting ticket from the Kerberos server.

-l ***lifetime*** (duration string.) Specifies the lifetime to be requested for the ticket; if this option is not specified, the default ticket lifetime (12 hours) is used instead.

-r ***time*** (duration string.) Specifies that the **renewable** option should be requested for the ticket, and specifies the desired total lifetime of the ticket.

-p specifies that the **proxiable** option should be requested for the ticket.

-f option specifies that the **forwardable** option should be requested for the ticket.

-e ***command*** **[*args* ...]** ksu proceeds exactly the same as if it was invoked without the **-e** option, except instead of executing the target shell, ksu executes the specified command. Example of usage:

```
ksu bob -e ls -lag
```

The authorization algorithm for **-e** is as follows:

If the source user is root or source user == target user, no authorization takes place and the command is executed. If source user id != 0, and `~target_user/.k5users` file does not exist, authorization fails. Otherwise, `~target_user/.k5users` file must have an appropriate entry for target principal to get authorized.

The .k5users file format:

A single principal entry on each line that may be followed by a list of commands that the principal is authorized to execute. A principal name followed by a * means that the user is authorized to execute any command. Thus, in the following example:

```
jqpublic@USC.EDU ls mail /local/kerberos/klist
jqpublic/secure@USC.EDU *
jqpublic/admin@USC.EDU
```

jqpublic@USC.EDU is only authorized to execute ls, mail and klist commands. jqpublic/secure@USC.EDU is authorized to execute any command. jqpublic/admin@USC.EDU is not authorized to execute any command. Note, that jqpublic/admin@USC.EDU is authorized to execute the target shell (regular ksu, without the **-e** option) but jqpublic@USC.EDU is not.

The commands listed after the principal name must be either a full path names or just the program name. In the second case, **CMD_PATH** specifying the location of authorized programs must be defined at the compilation time of ksu. Which command gets executed?

If the source user is root or the target user is the source user or the user is authorized to execute any command (* entry) then command can be either a full or a relative path leading to the target program. Otherwise, the user must specify either a full path or just the program name.

-a *args* Specify arguments to be passed to the target shell. Note that all flags and parameters following -a will be passed to the shell, thus all options intended for ksu must precede **-a**.

The **-a** option can be used to simulate the **-e** option if used as follows:

```
-a -c [command [arguments]].
```

-c is interpreted by the c-shell to execute the command.

4.6.9 INSTALLATION INSTRUCTIONS

ksu can be compiled with the following four flags:

GET_TGT_VIA_PASSWD In case no appropriate tickets are found in the source cache, the user will be prompted for a Kerberos password. The password is then used to get a ticket granting ticket from the Kerberos server. The danger of configuring ksu with this macro is if the source user is logged in remotely and does not have a secure channel, the password may get exposed.

PRINC_LOOK_AHEAD During the resolution of the default principal name, **PRINC_LOOK_AHEAD** enables ksu to find principal names in the .k5users file as described in the OPTIONS section (see **-n** option).

CMD_PATH Specifies a list of directories containing programs that users are authorized to execute (via .k5users file).

HAVE_GETUSERSHELL If the source user is non-root, ksu insists that the target user's shell to be invoked is a "legal shell". *getusershell(3)* is called to obtain the names of "legal shells". Note that the target user's shell is obtained from the passwd file.

Sample configuration:

```
KSU_OPTS = -DGET_TGT_VIA_PASSWD -DPRINC_LOOK_AHEAD -DCMD_PATH='"/bin /usr/ucb /local/
↪bin"
```

ksu should be owned by root and have the set user id bit turned on.

ksu attempts to get a ticket for the end server just as Kerberized telnet and rlogin. Thus, there must be an entry for the server in the Kerberos database (e.g., host/nii.isi.edu@ISI.EDU). The keytab file must be in an appropriate location.

4.6.10 SIDE EFFECTS

ksu deletes all expired tickets from the source cache.

4.6.11 AUTHOR OF KSU

GENNADY (ARI) MEDVINSKY

4.7 kswitch

4.7.1 SYNOPSIS

kswitch {**-c** *cachename*|**-p** *principal*}

4.7.2 DESCRIPTION

kswitch makes the specified credential cache the primary cache for the collection, if a cache collection is available.

4.7.3 OPTIONS

-c *cachename* Directly specifies the credential cache to be made primary.

-p *principal* Causes the cache collection to be searched for a cache containing credentials for *principal*. If one is found, that collection is made primary.

4.7.4 ENVIRONMENT

kswitch uses the following environment variables:

KRB5CCNAME Location of the default Kerberos 5 credentials (ticket) cache, in the form *type*:*residual*. If no *type* prefix is present, the **FILE** type is assumed. The type of the default cache may determine the availability of a cache collection; for instance, a default cache of type **DIR** causes caches within the directory to be present in the collection.

4.7.5 FILES

`FILE:/tmp/krb5cc_%{uid}` Default location of Kerberos 5 credentials cache

4.7.6 SEE ALSO

kinit, *kdestroy*, *klist*), kerberos(1)

4.8 kvno

4.8.1 SYNOPSIS

kvno [**-c** *ccache*] [**-e** *etype*] [**-q**] [**-h**] [**-P**] [**-S** *sname*] [**-U** *for_user*] *service1 service2 ...*

4.8.2 DESCRIPTION

kvno acquires a service ticket for the specified Kerberos principals and prints out the key version numbers of each.

4.8.3 OPTIONS

-c *ccache* Specifies the name of a credentials cache to use (if not the default)

-e *etype* Specifies the enctype which will be requested for the session key of all the services named on the command line. This is useful in certain backward compatibility situations.

-q Suppress printing output when successful. If a service ticket cannot be obtained, an error message will still be printed and kvno will exit with nonzero status.

-h Prints a usage statement and exits.

-P Specifies that the *service1 service2 ...* arguments are to be treated as services for which credentials should be acquired using constrained delegation. This option is only valid when used in conjunction with protocol transition.

-S *sname* Specifies that the *service1 service2 ...* arguments are interpreted as hostnames, and the service principals are to be constructed from those hostnames and the service name *sname*. The service hostnames will be canonicalized according to the usual rules for constructing service principals.

-U *for_user* Specifies that protocol transition (S4U2Self) is to be used to acquire a ticket on behalf of *for_user*. If constrained delegation is not requested, the service name must match the credentials cache client principal.

4.8.4 ENVIRONMENT

kvno uses the following environment variable:

KRB5CCNAME Location of the credentials (ticket) cache.

4.8.5 FILES

`FILE:/tmp/krb5cc_%{uid}` Default location of the credentials cache

4.8.6 SEE ALSO

kinit, *kdestroy*

4.9 sclient

4.9.1 SYNOPSIS

sclient *remotehost*

4.9.2 DESCRIPTION

sclient is a sample application, primarily useful for testing purposes. It contacts a sample server sserver(8) and authenticates to it using Kerberos version 5 tickets, then displays the server's response.

4.9.3 SEE ALSO

kinit, sserver(8)